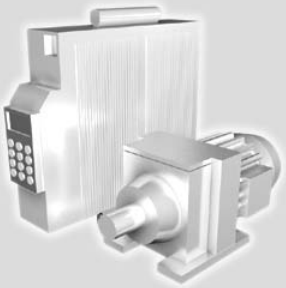
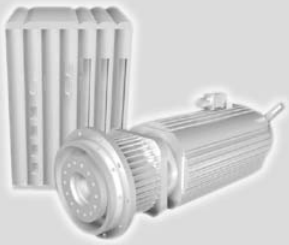
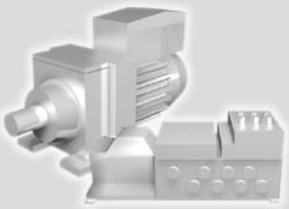
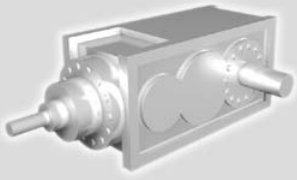
SEW EURODRIVE









Controle Seqüencial e de Posicionamento IPOS$^{plus®}$

FE310000

Edição 08/2007

1132 0583 / BP

Manual

SEW EURODRIVE

1 Indicações Importantes 7

2 Descrição do Sistema 8
2.1 Introdução 8
2.2 Características do IPOS$^{plus®}$ 9
2.3 Controle do IPOS$^{plus®}$ 11
2.4 Versão Aplicação / Módulos Aplicativos 12
2.5 Dados Técnicos 14
2.6 Documentos de referência 15

3 Variáveis IPOS 16
3.1 Introdução 16
3.2 Resumo das variáveis do sistema 17

4 Gerenciamento do Task e Interrupções 26
4.1 Introdução 26
4.2 Gerenciamento do Task para MOVIDRIVE® A e B 26
4.3 Tarefas para MOVIDRIVE® A 29
4.4 Tarefas para MOVIDRIVE® B 29
4.5 Interrupções 31
4.6 Interrupções para MOVIDRIVE® A e B 31
4.7 Interrupções variáveis com MOVIDRIVE® B 34

5 Detecção da Posição e Posicionamento 38
5.1 Introdução 38
5.2 Avaliação do encoder 38
5.3 Combinações do encoder 39
5.4 Encoder externo (X14) 41
5.5 Encoder absoluto SSI (DIP) 44
5.6 Referenciamento 47
5.7 Função Módulo 58
5.8 Controladores de came eletrônico 70

6 IPOS$^{plus®}$ e Fieldbus 80
6.1 Introdução 80
6.2 Entradas e saídas digitais 81
6.3 Dados cíclicos de processo 81
6.4 Comunicação acíclica 83
6.5 Características especiais de comunicação através do SBus 83
6.6 Palavras de controle fieldbus e palavras de estado fieldbus 83

7 IPOS$^{plus®}$ e Movimentos Sincronizados 87
7.1 Introdução 87
7.2 Sincronização da rotação através da função mestre/escravo 87
7.3 Operação síncrona com placa opcional DRS 87
7.4 Operação síncrona com a versão aplicação "Operação síncrona interna" 94
7.5 Operação síncrona com a versão aplicação "Came Eletrônico" 95

8 IPOS$^{plus®}$ para MQx 96
8.1 Introdução 96
8.2 Iniciando a ferramenta de programação 96
8.3 Sistema de controle seqüencial 96
8.4 Entradas e saídas digitais 97
8.5 Avaliação do sensor de proximidade 98
8.6 Contador 99

8.7 Valores da variável DIAG11 para a irregularidade IPOS ILLOP 99

9 Parâmetros IPOS$^{plus®}$ 101
9.1 Introdução 101
9.2 P90x Referenciamento IPOS 101
9.3 P91_ Parâmetros de deslocamento IPOS 104
9.4 P92_ Monitoração IPOS 109
9.5 P93x Funções especiais IPOS 110
9.6 P94x Encoder IPOS 112
9.7 P95x DIP 115
9.8 P96x Função módulo IPOS$^{plus®}$ 117

10 Compiler – Editor 118
10.1 Características técnicas 118
10.2 Primeira fase 118
10.3 Compiler – editor 130
10.4 Ajustes para o compilador IPOS$^{plus®}$ 132
10.5 Função de busca 135
10.6 Criando um novo projeto 137
10.7 Salvando um projeto 140
10.8 Elaboração da estrutura de gerenciamento do projeto 141
10.9 Abrindo um projeto 142
10.10 Manuseio do projeto com MOVIDRIVE® B 142
10.11 Compilando um projeto 143
10.12 Compilando e realizando o download 145
10.13 Iniciando um programa 145
10.14 Parando um programa 145
10.15 Comparar com o conversor 146
10.16 Debugger 146
10.17 Janela de variáveis 147
10.18 Informação do programa 149
10.19 Inserindo comandos 150
10.20 Comentários 151
10.21 Resumo dos ícones 152

11 Compiler – Programação 153
11.1 Pré-processador 154
11.2 Comandos do pré-processador 155
11.3 #include 156
11.4 Incluir pastas 157
11.5 #define 157
11.6 #undef 158
11.7 #declare 159
11.8 Estruturas padrão SEW 160
11.9 Estruturas definidas pelo usuário 166
11.10 long 168
11.11 initial long 168
11.12 #pragma 169
11.13 Explicação de const.h and io.h / constb.h and iob.h 170
11.14 Identificadores 171
11.15 Constantes 172
11.16 Variáveis IPOS$^{plus®}$ no compilador 172
11.17 Declaração das variáveis globais 172
11.18 Endereçamento indireto – indicador 173
11.19 numof() 174

12 Compiler – Operadores 175
12.1 Ordem de prioridade dos operadores 175
12.2 Operadores Unary 176
12.3 Operadores Binary 177
12.4 Operadores Ternary 177

13 Compiler – Construções 178
13.1 if...else 178
13.2 for 179
13.3 while 181
13.4 do...while 182
13.5 switch...case...default 184
13.6 return 184

14 Compiler – Funções 186
14.1 Funções definidas pelo usuário 186
14.2 Resumo dos comandos para funções padrão 187
14.3 Funções padrão 189

15 Compiler – Exemplos 224
15.1 Ajuste dos bits e dos bornes de saída 224
15.2 Desligamento dos bits e dos bornes de saída 225
15.3 Verificação dos bits e dos bornes de entrada 226
15.4 Verificação de um pulso 227
15.5 Valor absoluto de um número 229
15.6 Comando MoviLink 229
15.7 Comunicação SCOM 232
15.8 Processamento de interrupção do touch probe 234
15.9 Máquina de estado, controle fieldbus com modo de emergência 236
15.10 Estrutura de programação do compilador 239

16 Mensagens de Erro 246
16.1 Mensagens de erro do compilador 246

17 Assembler – Introdução 247
17.1 Características técnicas 247
17.2 Ajuste das unidades de deslocamento do usuário 247
17.3 Primeira fase 250

18 Assembler – Editor 254
18.1 Criando programas 256
18.2 Compilando e efetuando download 257
18.3 Iniciando / parando programas 258
18.4 Comparação entre arquivo e conversor 258
18.5 Debugger 259
18.6 Carregando o programa a partir do conversor 259
18.7 Resumo dos ícones 260

19 Assembler – Programação 261
19.1 Básicos 261
19.2 Entradas/saídas digitais 263
19.3 Entradas/saídas analógicas 267

20 Assembler – Comandos 269
20.1 Resumo dos comandos 269
20.2 Comandos aritméticos 273
20.3 Comandos do Bit 278
20.4 Comandos de comunicação 279
20.5 Comandos de posicionamento 296
20.6 Comandos do programa 302
20.7 Comandos de ajuste 309
20.8 Comandos especiais 321
20.9 Comandos de comparação 326

21 Assembler – Exemplos 332
21.1 Exemplo do programa "Flashing light" 332
21.2 Exemplo do programa "hoist" 334
21.3 Exemplo do programa "Jog mode" 340
21.4 Exemplo do programa "Table positioning" 343

22 Índice de Alterações 347

23 Índice Alfabético 348

Ref.: IPOS$^{plus®}$ Positioning and Sequence Control - Manual - Edition 11/2004 (11320419/EN)

1 Indicações Importantes

Ler este manual cuidadosamente antes de instalar e colocar em operação o IPOS$^{plus®}$.

Com este manual o usuário se familiariza com a documentação do IPOS$^{plus®}$.

Indicações de Segurança

Seguir sempre as instruções de segurança e aviso contidos neste manual.

Observar atentamente as seguintes indicações de segurança:

Risco de choque elétrico, por exemplo, ao trabalhar durante a operação.

Risco mecânico, por exemplo, ao trabalhar em elevadores de carga.

Instruções importantes para operação segura e livre de problemas da máquina/sistema acionado, por exemplo, pré-ajuste antes da colocação em operação.

Informação importante.

O sistema de controle seqüencial e de posicionamento IPOS$^{plus®}$ permite o ajuste do IPOS$^{plus®}$ para atender às especificações exatas de sua aplicação. Como com todos os sistemas de posicionamento há, no entanto, o risco de um erro de programação, que pode resultar em um comportamento inesperado do sistema (embora não incontrolado).

A SEW-EURODRIVE fabrica e testa cada equipamento conforme padrões e especificações técnicas atuais SEW-EURODRIVE.

O fabricante reserva-se o direito de fazer qualquer mudança nos dados técnicos e projetos descritos nesta documentação, no interesse do progresso técnico.

Favor atentar às instruções de operação para garantir:

- Operação livre de problema
- Realização de qualquer reclamação dentro do período de garantia

2 Descrição do Sistema

2.1 Introdução

As funções básicas e opcionais do IPOS$^{plus®}$ asseguram que o programa não seja somente um controlador de rotação malha aberta.

Na verdade, o sistema de controle seqüencial e de posicionamento integrado ao MOVIDRIVE® pode freqüentemente auxiliar no alívio da carga de processamento do CLP, ou mesmo substituí-lo.

A redução do controle central oferece aos clientes SEW economia significante de potencial nas condições de hardware e na complexidade da instalação elétrica.

O trabalho de programação é dividido entre o CLP e o conversor. No entanto, os usuários devem familiarizar-se com o sistema. Isto inclui conhecer o IPOS$^{plus®}$ caso queira tornar eficaz o uso de seus benefícios.

Âmbito desta documentação

Esta documentação abrange os seguintes tópicos:

Primeiro, descreve as funções de linguagem independente do IPOS$^{plus®}$. Estas características correspondem às do "controlador de rotação malha aberta":

- Controle de posicionamento
- Processamento da posição
- Gerenciamento de tarefa
- Gerenciamento de interrupção
- Parâmetros IPOS
- Variáveis IPOS

A seguir, a programação é descrita na linguagem assembler. Esta linguagem foi utilizada no passado para MOVIDYN® e MOVITRAC® 31C pelos programadores de CLP que já estavam habituados à programação em listas de instruções. A SEW-EURODRIVE recomenda a criação de novos programas na linguagem compiler. Todos os equipamentos MOVIDRIVE® podem ser programados na linguagem compiler.

Após o procedimento de programação na linguagem assembler ter sido descrito, este é seguido pela descrição de programação na linguagem compiler.

A seção final descreve os exemplos do programa. Isto inclui um exemplo para iniciantes de um programa seqüencial, com a estrutura básica do estado da máquina. Recomendamos iniciar com a estrutura básica e desenvolver o programa do usuário a partir daí.

MQx e MOVITRAC® 07

Nos módulos MQX do sistema descentralizado e nos conversores MOVITRAC® 07 LOGODrive, também é possível criar programas seqüenciais na linguagem compiler ou assembler. No entanto, neste caso o uso dos comandos é consideravelmente restrito. Esta documentação não detalha sobre o MOVITRAC® 07. As interfaces fieldbus MQx são descritas na seção "IPOS para MQx."

Criação de programas

Pode-se criar programas IPOS$^{plus®}$ utilizando o assembler ou o compiler. Ambas ferramentas de programação estão inclusas no pacote do programa MOVITOOLS®.

Os módulos aplicativos resolvem tarefas típicas do acionamento sem que o usuário tenha que criar um programa. Em vez de programar, você tem somente que ajustar os parâmetros para um programa (módulo aplicativo) criado pela SEW-EURODRIVE. Isto economiza tempo, e não é necessário a experiência de programação descrita neste manual.

2.2 Características do IPOS$^{plus®}$

- O **controle seqüencial** IPOS$^{plus®}$ possibilita rodar um programa do usuário, independente de qualquer realimentação do encoder ou do modo de controle selecionado (VFC, CFC, SERVO).
- Em conjunto com a realimentação do encoder, o **controle de posicionamento** IPOS$^{plus®}$ fornece capacidade de posicionamento ponto-a-ponto de alto desempenho.
- O programa roda independente da realimentação do encoder e do modo de operação.
- O MOVIDRIVE® continua a rodar o programa do usuário mesmo se o equipamento desenvolve um mau funcionamento (a solução de problemas é possível no programa do usuário)
- O IPOS$^{plus®}$ pode rodar vários programas do usuário / tarefas simultaneamente, independente de um outro. Tasks podem ser interrompidos utilizando interromper.
- Opcionais de controle total e de fácil utilização para IPOS$^{plus®}$.
- Acesso para todos os opcionais disponíveis:
  - Placa de entrada/saída
  - Interfaces fieldbus
  - Placa de operação em sincronismo
- Diversas possibilidades de comunicação:
  - System bus (S-bus)
  - RS-485
  - RS-232 (com conversores de interface USS21A, UWS11A, UWS21A)
  - Interfaces fieldbus
- Processamento dos sinais de entrada/saída digitais e analógicos.
- Posicionamento com velocidade de deslocamento ajustável e rampas de posicionamento.
- Pré-ajuste para controle de posição, rotação e torque com erro por atraso minimizado.
- Processamento de encoder absoluto.
- São disponíveis 1024 variáveis 32-bit no programa IPOS$^{plus®}$. Destas variáveis, 128 podem ser armazenadas protegidas contra falha de potência não voláteis.
- Com o IPOS$^{plus®}$, todos os parâmetros do conversor podem ser lidos e escritos através dos comandos de comunicação.
- 2 entradas touch probe.
- Tipos de rampa:
  - Linear
  - Senoidal
  - Quadrática
  - Rampa via rede
  - Suavização de picos
  - Came eletrônico
  - I-operação em sincronismo angular

- Funções de estado e controle:
  - Controle de erro por atraso
  - Monitoração de posição
  - Chaves fim-de-curso do software e do hardware
  - Função do encoder
- 8 tipos de referenciamento.
- As funções a seguir podem ser mudadas durante o movimento:
  - Posição final
  - Velocidade de deslocamento
  - Rampa de posicionamento
  - Torque
- Possibilidade de "posicionamento contínuo."
- Função Override (ajuste da velocidade de posicionamento pela entrada analógica).
- As funções tecnológicas a seguir podem ser controladas com um encoder virtual:
  - Came eletrônico
  - Operação em sincronismo interno
- A programação no compiler também oferece:
  - Criação do programa em uma linguagem de alto nível (linguagem C)
  - Nomes simbólicos das variáveis
  - Possibilidade de criar módulos de programa que podem ser utilizados em outros projetos
  - Programação clara, modular e estruturada
  - Controle compiler utilizando comandos pré-processadores
  - Estruturas padrão
  - Estruturas definidas pelo usuário
  - Funções padrão disponíveis
  - Debugger para solução de problemas
  - Amplos opcionais para comentários (os comentários são armazenados somente no PC)
- A programação no assembler oferece:
  - Linhas de comentários
  - A programação nas unidades de deslocamento pode ser definida pelo usuário (as unidades são colocadas no cabeçalho do programa)

- Seleção do setpoint. Dependendo do hardware e do setpoint necessário, são disponíveis os seguintes opcionais para a especificação:
  - Setpoint analógico
  - Setpoint fixo
  - Setpoint fixo + setpoint analógico
  - Potenciômetro do motor
  - Operação mestre/escravo com SBus
  - Operação mestre/escravo com RS-485
  - Setpoint DRS (somente com o opcional DRS11A)
  - Setpoint fieldbus/fieldbus monitor (somente com a interface fieldbus opcional)
  - Setpoint de posição IPOS$^{plus®}$

  Caso seja necessário utilizar realimentação do encoder para processamento do valor de ajuste, depende do modo de operação que é selecionado. O setpoint ativo depende dos seguintes ajustes:
  - Modo de operação P700
  - Fonte do setpoint P100
  - Ajustes de parâmetro dos bornes de entrada P600 ... P619
  - Seleção/modo motor dos dados PO
  - Seleção da operação manual

2.3 Controle do IPOS$^{plus®}$

O IPOS$^{plus®}$ pode ser controlado conforme segue:

- Controle através dos bornes de entrada no equipamento
- Ajuste da palavra de controle IPOS$^{plus®}$ para "variável do sistema" H484 CTRL. WORD
- Interface RS-485
- Interface fieldbus
- SBus (system bus)

Ativa a fonte do sinal de controle

O controle através dos bornes de entrada e da palavra de controle IPOS$^{plus®}$ H484 está sempre ativo.

Pode-se determinar fontes adicionais do sinal de controle utilizando os seguintes parâmetros:

- Fonte do setpoint P100
- Fonte do sinal de controle P101
- Descrição dos dados de processo P870 ... P872

2.4 Versão aplicação / Módulos aplicativos

Versão aplicação

O MOVIDRIVE® com versão aplicação (terminação OT na denominação) oferece funções adicionais, como:

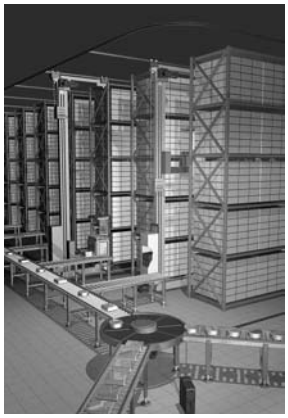
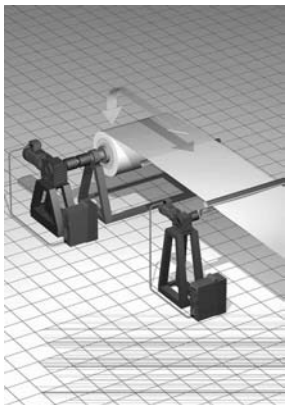
- Operação em sincronismo interno (I-Sync)
- Came eletrônico
- Módulos aplicativos
- Auto ASR (Ajuste anti escorregamento, atualmente disponível somente com MOVIDRIVE® A)
- SBus TP (SBus touch probe, disponível somente com MOVIDRIVE® A; no equipamento padrão MOVIDRIVE® B, isto é solucionado utilizando uma interrupção variável)
- Faca rotativa (cross cutter, em preparação para MOVIDRIVE® B)

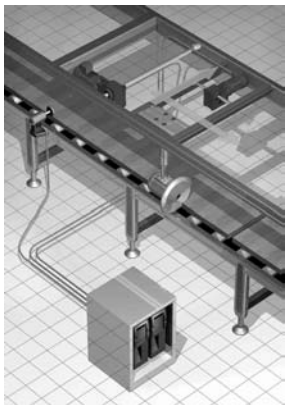
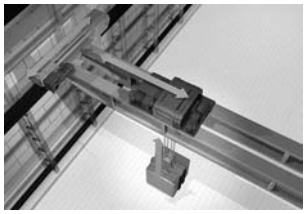
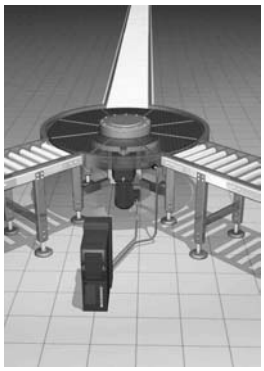
As funções "operação em sincronismo interno" e "came eletrônico" são explicadas na seção "IPOSplus® e movimentos sincronizados" e descritas detalhadamente em manuais separados. Nestes casos, as variáveis IPOSplus® adicionais são funções atribuídas do sistema que podem ser endereçadas nos programas IPOSplus® do usuário.

Módulos aplicativos

Um módulo aplicativo é um programa protegido de usuário, projetado pela SEW, que pode ser carregado no conversor.

Um pacote detalhado de funções coordenadas, telas de entrada de fácil utilização e simples orientação ao usuário simplificam a colocação em operação. O usuário não pode alterar o programa IPOSplus®.

Resumo dos módulos aplicativos	
Posicionamento	
Bobinador / Desbobinador	

Resumo dos módulos aplicativos	
Serra móvel	
Operação em sincronismo I-SYNC (somente para MOVIDRIVE® B e MCH)	
Posicionamento giratório	

Os módulos aplicativos inteligentes na versão aplicação oferecem um novo nível de funcionalidade. Todos os dados importantes da máquina são facilmente acessíveis. Quase não há possibilidades para erros, já que somente os parâmetros necessários para a aplicação são inseridos. Todos os dados relacionados, por exemplo, estados do borne ou valores de posição, podem ser observados utilizando uma ferramenta de diagnósticos durante o processo de operação.

A funcionalidade de cada um destes módulos é descrita em manuais individuais.

2.5 Dados técnicos

Número de pulsos do encoder por volta: MOVIDRIVE® X15, encoder do motor MOVIDRIVE® X62, encoder síncrono MOVIDRIVE® X62, encoder absoluto (incluindo encoder absoluto HIPERFACE®)	O IPOS$^{plus®}$ sempre opera com 4096 incrementos / volta do motor (Pré-requisito: Volta do encoder de 512, 1024 ou 2048 pulsos / volta do motor (não são permitidas outras voltas do encoder) ou resolver)
Comprimento máximo do programa / memória do programa:	MOVIDRIVE® A (sem MCH): 4 kByte MOVIDRIVE® MCH: 8 kByte MOVIDRIVE® B: 16 kByte 1 kByte corresponde a aprox. 200 ... 250 comandos assembler
Tempo de processamento do comando:	MOVIDRIVE® A: Tarefa 1: 1 comando assembler / ms Tarefa 2: 2 comandos assembler / ms MOVIDRIVE® B (o total da tarefa 1 e da tarefa 2 ≤ 12 comandos assembler / ms): Tarefa 1: 1 ... 10 comandos assembler / ms Tarefa 2: 2 ... 11 comandos assembler / ms Tarefa 3: Tempo de processamento aberto
Interrupções:	MOVIDRIVE® A: 1 interrupção ativada por vez, erro ou touch probe interrompe tarefa 1. MOVIDRIVE® B: Como MOVIDRIVE® A; 4 variáveis de interrupção adicionais que interrompem tarefa 2 e tarefa 3.
Variáveis:	MOVIDRIVE® A: 512 / MOVIDRIVE® B: 1024, disso 128 (0 ... 127) podem ser armazenados na memória não-volátil. Exceções: MDS, MDV, MCS, MCV com o opcional came eletrônico. Neste caso, 16 variáveis (0 ... 15) podem ser armazenadas na memória não-volátil. Faixa de valor: -2^{31} ... $+(2^{31}-1)$
Entradas Touch probe:	2 entradas, tempo de processamento 200 µs
Intervalo de amostragem das entradas analógicas:	1 ms
Intervalo de amostragem das entradas digitais:	MOVIDRIVE® A: 5 ms MOVIDRIVE® B: 1 ms
Entradas/saídas digitais:	MOVIDRIVE® A: 6 entradas / 3 saídas MOVIDRIVE® B: 8 entradas / 6 saídas Opcional DIO: 8 entradas / 8 saídas Opcional DIP: 8 entradas / 8 saídas
Entradas/saídas analógicas:	MOVIDRIVE®: 1 entrada (0 ... 10 V, ± 10 V, 0 ... 20 mA, 4 ... 20 mA) Opcional DIO: 1 entrada (0 ... 10 V, ± 10 V, 0 ... 20 mA) 2 saídas (± 10 V, 0 ... 20 mA, 4 ... 20 mA)

2.6 Documentos de referência

Este documento descreve o sistema integrado de controle seqüencial e de posicionamento IPOS$^{plus®}$ no MOVIDRIVE®.

A lista de referência a seguir é um resumo dos documentos relacionados neste manual. Não é necessário ler estes documentos para programar com IPOS$^{plus®}$; eles simplesmente oferecem informação adicional.

Todos os documentos estão disponíveis na website da SEW-EURODRIVE em http://www.sew-eurodrive.com

Manuais gerais

- Manual de Sistema do MOVIDRIVE® compact
- Manual de Sistema do MOVIDRIVE® MD_ 60A
- Manual de Sistema do MOVIDRIVE® MD_ B

Manuais para interfaces seriais / fieldbuses

- Comunicação Serial MOVIDRIVE®
- MOVIDRIVE® Protocolo Fieldbus com Lista de Parâmetros
- MOVIDRIVE® Interface Fieldbus DFP21
- MOVIDRIVE® Interface Profibus DFP11
- MOVIDRIVE® Interface Interbus DFI11
- MOVIDRIVE® Interface Fieldbus Interbus DFI21 com Cabo de Fibra Óptica
- MOVIDRIVE® Interface DeviceNet DFD11
- MOVIDRIVE® Interface CAN Bus DFC11
- MOVIDRIVE® Interface Fieldbus DF011 CANopen

Manuais para movimentos sincronizados do eixo

- MOVIDRIVE® Came Eletrônico, adendo ao manual de sistema
- MOVIDRIVE® Placa de Operação em Sincronismo Tipo DRS11
- MOVIDRIVE® Operação em Sincronismo Interno

Manuais para módulos aplicativos

- MOVIDRIVE® Posicionamento com Opcional Encoder Absoluto DIP11
- MOVIDRIVE® Posicionamento Extendido através da Rede
- MOVIDRIVE® Rede de Posicionamento
- MOVIDRIVE® Tabela de Posicionamento com Controle da Rede
- MOVIDRIVE® Módulo de Posicionamento

Manuais para as interfaces fieldbus MQx

- Sistema para Instalação Descentralizada: Interfaces PROFIBUS, Distribuidores de Campo
- Sistema para Instalação Descentralizada: Interfaces INTERBUS, Distribuidores de Campo
- Sistema para Instalação Descentralizada: Interfaces DeviceNet/CANopen, Distribuidores de Campo

3 Variáveis IPOS

3.1 Introdução

O sistema integrado de controle seqüencial e de posicionamento utiliza variáveis globais que são utilizadas por todas as tarefas e interrupções. **Não** há variáveis locais que são declaradas somente em uma tarefa ou em uma função.

São disponíveis as seguintes variáveis:

- MOVIDRIVE® A: 512 variáveis H0 ... H511
- MOVIDRIVE® B: 1024 variáveis H0 ... H1023

No MOVIDRIVE® A e no MOVIDRIVE® B, são armazenadas na memória não-volátil 128 variáveis (H0 ... H127) protegidas contra falha de potência. Exceções: MDS, MDV, MCS, MCV com o opcional came eletrônico. Neste caso, 16 variáveis (H0 ... H15) podem ser armazenadas protegidas contra falha de potência.

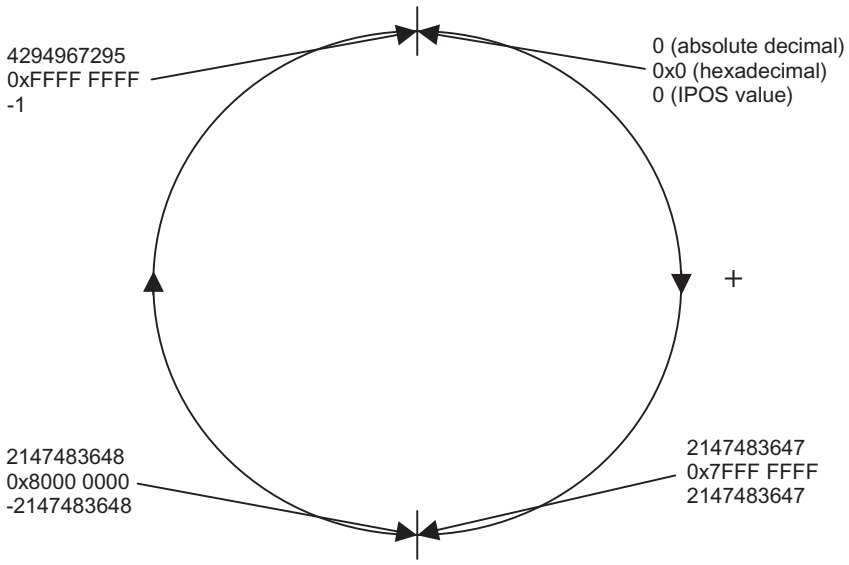
Todas as variáveis são de 32-bit tratadas como números inteiros durante cálculos e comparações. No programa do usuário, deve-se verificar que o resultado final de um cálculo encontra-se dentro de uma determinada faixa de números.

Exemplo

$$H0 = 4, \quad H1 = 7, \quad H3 = \frac{H0}{H7} = 0$$

$$H0 = 2147483647, \quad H1 = 1, \quad H3 = H0 + H1 = -2147483648$$

A faixa de número pode ser representada em um círculo, conforme a seguir:



Cada variável tem um índice que pode ser utilizado para ler e escrever variáveis utilizando, por exemplo, o comando Movilink. O índice é calculado conforme a seguir:

Index = VarNo. + 11000

Exemplo: H371 tem o índice 11371.

3.2 Resumo das variáveis do sistema

Algumas variáveis IPOS são atribuídas às funções de ajuste e são submetidas como variáveis do sistema.

São elas:

- Para MOVIDRIVE® A, as variáveis IPOS H453 até H511
- Para MOVIDRIVE® B, as variáveis IPOS H453 até H560

Os nomes simbólicos são disponíveis no compiler quando uma das seguintes linhas é inserida no início do programa:

```
#include <const.h>   //symb. names system variables MOVIDRIVE A
#include <constb.h>  //symb. names system variables MOVIDRIVE B
```

A tabela a seguir descreve a função das variáveis do sistema e seus nomes no compiler e assembler.

As variáveis que não são atribuídas na faixa especificada, são reservadas para funções internas e não podem ser utilizadas para variáveis do usuário.

Nr.	Nome Compiler / assembler	Descrição
128		Esta variável pode ser utilizada em um programa IPOS específico do usuário. A variável é utilizada pelos módulos de aplicação para armazenar a identificação do programa.
360 ... 450	Faixa de variáveis para operação em sincronismo interno ou came eletrônico	Esta faixa de variáveis é atribuída às variáveis adicionais do sistema se forem utilizados opcionais de operação em sincronismo interno ou came eletrônico. Em todos os outros casos, estas podem ser utilizadas pelo usuário quando necessário.
453	MóduloCtrl / MODULOCTRL	Palavra de controle para a função módulo (ver também função módulo e parâmetro IPOS). **Bit 0 TargetReset_Off** Bit 0 = 0: A posição final é presa mesmo se a liberação tiver sido revogada ou se o regulador bloqueado ou o bit de parada tiver sido ajustado. Se o conversor estiver novamente liberado, ele continua com o processo de posicionamento. Bit 0 = 1: A tarefa de posicionamento atual é eliminada (ModTagPos é ajustado para ModActPos) se a operação de posicionamento for interrompida (por exemplo, se a liberação for revogada ou se o regulador bloqueado ou o bit de parada estiver ajustado). **Bit 1 TargetGAZ_Select** Bit 1 = 0: Ajuste padrão, saída 360° corresponde a 2^{16} incr. Bit 1 = 1: Ajuste para aumentar a volta: 360° corresponde ao produto do módulo numerador P961 x módulo número de pulsos do encoder P963. O posicionamento não pode ser realizado em diversas voltas.
454	ModTagPos / MOD.TAGPOS	Módulo Posição Final Se um valor alterado for escrito na posição final do módulo para um conversor liberado, o posicionamento começa no eixo de saída. O valor nominal de posição (para H453.1 = 0) é ajustado em 16 bit no equipamento H454 MODTAGPOS = k x 360° + 0 ... 360°= k x 2^{16} + 0 ... (2^{16} –1) (k = número de voltas completas). Uma vez o novo valor ter sido escrito para a variável, somente a posição final dentro de uma volta é visível na variável H454. Recomendamos também escrever o novo valor para uma variável temporária para um melhor diagnóstico. Uma vez a posição 454 ter sido escrita, o firmware calcula um alvo incremental H492. Isto faz com que o bit 19 da H473 "In position" permaneça em nível 1 por até 1 ms.
455	ModActPos / MOD.ACTPOS	Módulo Posição Atual O módulo posição atual movimenta (na resolução 16 bit quando H453.1 = 0) entre 0 e 2^{16} incrementos (0° e 360°).
456	ModCount / MOD COUNT	Incrementos dentro da volta do módulo antes de refletir para a saída. Indica o valor do resultado temporário interno quando o valor do encoder incremental H509 / H510 / H511 (IPOS valor do encoder) é convertido para a posição atual do módulo H455. Para H456 = (IPOS valor do encoder P962) MOD (P961 x P963) H455 = H456 / (P961 x P963) x 2^{16} (pré-requisito: H453, bit 1 = 0) Ver o capítulo "Módulo de Posicionamento". Se 0 for escrito para H456, H455 é ajustado automaticamente para 0.

<table>
<tr><th>Nr.</th><th>Nome
Compiler /
assembler</th><th>Descrição</th></tr>
<tr><td>473</td><td>StatusWord /
STAT.WORD</td><td>A palavra de estado pode ser utilizada para questionar o estado de operação do conversor.</td></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Bit</th><th>Função no nível "1"</th><th>Bit</th><th>Função no nível "1"</th></tr>
<tr><td>0</td><td>Sem função</td><td>13</td><td>Sinal Imax (P442)</td></tr>
<tr><td>1</td><td>/Irregularidade</td><td>14</td><td>/Utilização do motor 1</td></tr>
<tr><td>2</td><td>Pronto</td><td>15</td><td>/Utilização do motor 2</td></tr>
<tr><td>3</td><td>Estágio de saída lig</td><td>16</td><td>/DRS pré-aviso</td></tr>
<tr><td>4</td><td>Campo girante lig</td><td>17</td><td>/DRS erro por atraso</td></tr>
<tr><td>5</td><td>Freio liberado</td><td>18</td><td>DRS escravo em posição</td></tr>
<tr><td>6</td><td>Freio atuado</td><td>19</td><td>IPOS em posição (ver também H493)</td></tr>
<tr><td>7</td><td>Motor em parada (de n < 20 rpm)</td><td>20</td><td>IPOS referenciado</td></tr>
<tr><td>8</td><td>Jogo de parâmetro</td><td>21</td><td>Reservado</td></tr>
<tr><td>9</td><td>Referência de rotação (P400)</td><td>22</td><td>/Irregularidade IPOS</td></tr>
<tr><td>10</td><td>Janela de rotação (P410)</td><td>23..31</td><td>Reservado</td></tr>
<tr><td>11</td><td>Comparação valor de ajuste/atual (P410)</td><td></td><td></td></tr>
<tr><td>12</td><td>Referência atual (P430)</td><td></td><td></td></tr>
</table>

A saída "IPOS in position" também é ajustada quando a liberação é revogada ou o regulador bloqueado é ajustado.
</td></tr>
<tr><td>474</td><td>Scope474 /
SCOPE 474</td><td rowspan="2">Estas duas variáveis podem ser utilizadas junto com a função osciloscópio SCOPE integrada no MOVI-TOOLS para gravar os valores medidos.
Exemplo: Medição do valor atual de posição de um módulo de eixo. No programa IPOS, o comando H474 = H455 é acessado ciclicamente e no SCOPE, o canal 1 é ajustado para variável IPOS H474 Low e o canal 2 é ajustado para variável IPOS H474 High.</td></tr>
<tr><td>475</td><td>Scope475 /
SCOPE 475</td></tr>
<tr><td>476</td><td>DRS_Ctrl /
DRS CTRL.</td><td>Nível de sinal das saídas digitais da placa de operação em sincronismo DRS11, READ e SET.
<table>
<tr><th>Bit</th><th>Nível do borne</th></tr>
<tr><td>0</td><td>X40.9 OUTP0</td></tr>
<tr><td>1</td><td>X40.10 OUTP1</td></tr>
<tr><td>2..14</td><td>Reservado</td></tr>
<tr><td>15</td><td>Ajustar irregularidade DRS do hardware (irregularidade 48)</td></tr>
<tr><td>16..31</td><td>Reservado</td></tr>
</table>
</td></tr>
<tr><td>477</td><td>DRS_Status /
DRS STATUS</td><td>Nível de sinal das entradas digitais e sinais de estado da placa de operação em sincronismo tipo DRS11, READ.
<table>
<tr><th>Bit</th><th>Nível do borne / sinais de estado</th></tr>
<tr><td>0</td><td>X40.5 INP4 Entrada livre 1</td></tr>
<tr><td>1</td><td>X40.6 INP5 Entrada livre 2</td></tr>
<tr><td>2</td><td>/DRS pré-aviso</td></tr>
<tr><td>3</td><td>/DRS erro por atraso</td></tr>
<tr><td>4</td><td>Escravo DRS em posição</td></tr>
<tr><td>5</td><td>Mestre em parada</td></tr>
<tr><td>6..31</td><td>Reservado</td></tr>
</table>
</td></tr>
<tr><td>478</td><td>AnaOutpIPOS2 /
ANA.OUT IP2</td><td>Saídas analógicas da placa de expansão do borne tipo DIO11, somente SET.
O valor da variável H478 é de uma saída analógica quando o borne correspondente é programado para "IPOS OUTPUT 2".
Opcional DIO11 é necessário para MOVIDRIVE® A e B; para MCH e MCS / MCV / MCV 40A, uma saída pode ser programada como saída digital ou saída analógica.
<table>
<tr><th>Valor variável</th><th>Saída física</th><th>Programação dos bornes de saída</th></tr>
<tr><td>– 10000..0..10000</td><td>AOV1/AOC1/AO01</td><td>P640 Saída analógica AO1 = IPOS OUTPUT 2</td></tr>
<tr><td>– 10000..0..10000</td><td>AOV2/AOC2/AO01</td><td>P643 Saída analógica AO2 = IPOS OUTPUT 2</td></tr>
</table>
</td></tr>
<tr><td>479</td><td>AnaOutpIPOS /
ANA.OUT IP</td><td>Saídas analógicas da placa de expansão do borne tipo DIO11, somente SET.
O valor da variável H479 é de uma saída analógica quando o borne correspondente é programado para "IPOS OUTPUT".
Opcional DIO11 é necessário para MOVIDRIVE® A e B; para MCH e MCS / MCV / MCV 40A, uma saída pode ser programada como saída digital ou saída analógica.
<table>
<tr><th>Valor variável</th><th>Saída física</th><th>Programação dos bornes de saída</th></tr>
<tr><td>– 10000..0..10000</td><td>AOV1/AOC1/AO01</td><td>P640 Saída analógica AO1 = IPOS OUTPUT</td></tr>
<tr><td>– 10000..0..10000</td><td>AOV2/AOC2/AO01</td><td>P643 Saída analógica AO2 = IPOS OUTPUT</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Nr.</th><th>Nome Compiler / assembler</th><th>Descrição</th></tr>
<tr><td>480</td><td>OptOutpIPOS / OPT.OUT IP</td><td>
Saídas digitais da placa de expansão do borne tipos DIO11 / DIP11, somente SET.
A função READ pode ser realizada para MOVIDRIVE® A utilizando H482 e para MOVIDRIVE® B utilizando H521.
Se o opcional DIO11 ou DIP11 não for inserido, os bornes virtuais podem ser ajustados na palavra de estado 2 através da rede se, por exemplo, P873 = STATUSWORD 2.
Os bits da variável H480 são reproduzidos nas saídas digitais da unidade básica se o borne relacionado for ajustado para IPOS OUTPUT.
Se uma saída digital é alternada no IPOS, a saída física no borne é alternada 1 ms depois, se for ajustado como saída IPOS.
<table>
<tr><th>Bit</th><th>Nome IPOS</th><th>DIO (+ DIO)</th><th>DIO + DIP</th><th>DIO + -field-bus</th><th>DIP</th><th>DIP + fieldbus</th><th>Fieldbus</th></tr>
<tr><td></td><td></td><td colspan="3">P63x é efetuado na DIO</td><td colspan="2">P63x é efetuado na DIO</td><td>P873 palavra de estado 2</td></tr>
<tr><td>0</td><td>DO10</td><td>X23:1</td><td>X23:1</td><td>X23:1</td><td>X61:1</td><td>X61:1</td><td>Bit 8</td></tr>
<tr><td>1</td><td>DO11</td><td>X23:2</td><td>X23:2</td><td>X23:2</td><td>X61:2</td><td>X61:2</td><td>Bit 9</td></tr>
<tr><td>2</td><td>DO12</td><td>X23:3</td><td>X23:3</td><td>X23:3</td><td>X61:3</td><td>X61:3</td><td>Bit 10</td></tr>
<tr><td>3</td><td>DO13</td><td>X23:4</td><td>X23:4</td><td>X23:4</td><td>X61:4</td><td>X61:4</td><td>Bit 11</td></tr>
<tr><td>4</td><td>DO14</td><td>X23:5</td><td>X23:5</td><td>X23:5</td><td>X61:5</td><td>X61:5</td><td>Bit 12</td></tr>
<tr><td>5</td><td>DO15</td><td>X23:6</td><td>X23:6</td><td>X23:6</td><td>X61:6</td><td>X61:6</td><td>Bit 13</td></tr>
<tr><td>6</td><td>DO16</td><td>X23:7</td><td>X23:7</td><td>X23:7</td><td>X61:7</td><td>X61:7</td><td>Bit 14</td></tr>
<tr><td>7</td><td>DO17</td><td>X23:8</td><td>X23:8</td><td>X23:8</td><td>X61:8</td><td>X61:8</td><td>Bit 15</td></tr>
<tr><td>8</td><td></td><td></td><td>X61:1</td><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td></td><td>(X23:1)</td><td>X61:2</td><td></td><td></td><td></td><td></td></tr>
<tr><td>10</td><td></td><td>...</td><td>X61:3</td><td></td><td></td><td></td><td></td></tr>
<tr><td>11</td><td></td><td>(X23:8)</td><td>X61:4</td><td></td><td></td><td></td><td></td></tr>
<tr><td>12</td><td></td><td></td><td>X61:5</td><td></td><td></td><td></td><td></td></tr>
<tr><td>13</td><td></td><td></td><td>X61:6</td><td></td><td></td><td></td><td></td></tr>
<tr><td>14</td><td></td><td></td><td>X61:7</td><td></td><td></td><td></td><td></td></tr>
<tr><td>15</td><td></td><td></td><td>X61:8</td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>481</td><td>StdOutpIPOS / STD.OUT IP</td><td>
Saídas digitais da unidade básica, somente SET.
Se uma saída digital é alternada no IPOS, a saída física no borne éalternada 1 ms depois, se for ajustado como saída IPOS.
<table>
<tr><th>Bit</th><th>Nome IPOS</th><th></th></tr>
<tr><td>0
(0)</td><td>DB00
DO00 MQx</td><td>Não pode ser programado, programação fixa com "/Brake"
Se P628 = IPOS OUTPUT (somente MQx)</td></tr>
<tr><td>1</td><td>DO01</td><td>Se P620 = IPOS OUTPUT</td></tr>
<tr><td>2</td><td>DO02</td><td>Se P621 = IPOS OUTPUT</td></tr>
<tr><td>3</td><td>DO03</td><td>Somente com MOVIDRIVE® B se P622 = IPOS OUTPUT</td></tr>
<tr><td>4</td><td>DO04</td><td>Somente com MOVIDRIVE® B se P623 = IPOS OUTPUT</td></tr>
<tr><td>5</td><td>DO05</td><td>Somente com MOVIDRIVE® B se P624 = IPOS OUTPUT</td></tr>
</table>
</td></tr>
<tr><td>482</td><td>OutputLevel / OUTPUT LVL MOVIDRIVE® A (MOVIDRIVE® B: H521)</td><td>
Nível de sinal das saídas digitais, somente LEITURA.
<table>
<tr><th>Bit</th><th>Nome IPOS</th><th>DIO (+ DIO)</th><th>DIO + DIP</th><th>DIO + -field-bus</th><th>DIP</th><th>DIP + fieldbus</th><th>Fieldbus</th></tr>
<tr><td></td><td></td><td colspan="3">P63x é efetuado na DIO</td><td colspan="2">P63x é efetuado na DIO</td><td>P873 palavra de estado 2</td></tr>
<tr><td>0</td><td>DB00</td><td>DB00</td><td>DB00</td><td>DB00</td><td>DB00</td><td>DB00</td><td>DB00</td></tr>
<tr><td>1</td><td>DO01</td><td>DO01</td><td>DO01</td><td>DO01</td><td>DO01</td><td>DO01</td><td>DO01</td></tr>
<tr><td>2</td><td>DO02</td><td>DO02</td><td>DO02</td><td>DO02</td><td>DO02</td><td>DO02</td><td>DO02</td></tr>
<tr><td>3</td><td>DO10</td><td>X23:1</td><td>X23:1</td><td>X23:1</td><td>X61:1</td><td>X61:1</td><td>Bit 8</td></tr>
<tr><td>4</td><td>DO11</td><td>X23:2</td><td>X23:2</td><td>X23:2</td><td>X61:2</td><td>X61:2</td><td>Bit 9</td></tr>
<tr><td>5</td><td>DO12</td><td>X23:3</td><td>X23:3</td><td>X23:3</td><td>X61:3</td><td>X61:3</td><td>Bit 10</td></tr>
<tr><td>6</td><td>DO13</td><td>X23:4</td><td>X23:4</td><td>X23:4</td><td>X61:4</td><td>X61:4</td><td>Bit 11</td></tr>
<tr><td>7</td><td>DO14</td><td>X23:5</td><td>X23:5</td><td>X23:5</td><td>X61:5</td><td>X61:5</td><td>Bit 12</td></tr>
<tr><td>8</td><td>DO15</td><td>X23:6</td><td>X23:6</td><td>X23:6</td><td>X61:6</td><td>X61:6</td><td>Bit 13</td></tr>
<tr><td>9</td><td>DO16</td><td>X23:7</td><td>X23:7</td><td>X23:7</td><td>X61:7</td><td>X61:7</td><td>Bit 14</td></tr>
<tr><td>10</td><td>DO17</td><td>X23:8</td><td>X23:8</td><td>X23:8</td><td>X61:8</td><td>X61:8</td><td>Bit 15</td></tr>
<tr><td>11</td><td></td><td></td><td>X61:1</td><td></td><td></td><td></td><td></td></tr>
<tr><td>12</td><td></td><td>(X23:1)</td><td>X61:2</td><td></td><td></td><td></td><td></td></tr>
<tr><td>13</td><td></td><td>...</td><td>X61:3</td><td></td><td></td><td></td><td></td></tr>
<tr><td>14</td><td></td><td>(X23:8)</td><td>X61:4</td><td></td><td></td><td></td><td></td></tr>
<tr><td>15</td><td></td><td></td><td>X61:5</td><td></td><td></td><td></td><td></td></tr>
<tr><td>16</td><td></td><td></td><td>X61:6</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td>X61:7</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td>X61:8</td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Nr.</th><th>Nome Compiler / assembler</th><th>Descrição</th></tr>
<tr><td>483</td><td>InputLevel / INPUT LVL MOVIDRIVE® A (MOVIDRIVE® B: H520)</td><td>Nível de sinal das entradas digitais, somente LEITURA.
<table>
<tr><th>Bit</th><th>Nome IPOS</th><th>DIO (+DIO)</th><th>DIO + DIP</th><th>DIO + -field-bus</th><th>DIP</th><th>DIP + fieldbus</th><th>Fieldbus P870 = palavra de estado 2</th></tr>
<tr><td>0
1
2
3
4
5</td><td>DI00
DI01
DI02
DI03
DI04
DI05</td><td colspan="6">Depende da unidade básica, por ex. X13:1
X13:2
X13:3
X13:4
X13:5
X13:6</td></tr>
<tr><td>6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21</td><td>DI10
DI11
DI12
DI13
DI14
DI15
DI16
DI17</td><td>X22:1
X22:2
X22:3
X22:4
X22:5
X22:6
X22:7
X23:8

(X23:1)
...
(X23:8)</td><td>X22:1
X22:2
X22:3
X22:4
X22:5
X22:6
X22:7
X22:8
X60:1
X60:2
X60:3
X60:4
X60:5
X60:6
X60:7
X60:8</td><td>X22:1
X22:2
X22:3
X22:4
X22:5
X22:6
X22:7
X22:8</td><td>X60:1
X60:2
X60:3
X60:4
X60:5
X60:6
X60:7
X60:8</td><td>X60:1
X60:2
X60:3
X60:4
X60:5
X60:6
X60:7
X60:8</td><td>Bit 8
Bit 9
Bit 10
Bit 11
Bit 12
Bit 13
Bit 14
Bit 15</td></tr>
</table></td></tr>
<tr><td>484</td><td>ControlWord / CTRL.WORD</td><td>Palavra de controle IPOS^{plus®} (funções do equipamento READ e SET).
A palavra de controle IPOS^{plus®} sempre pode ser utilizada, independente do modo de operação, fonte do sinal de controle e fonte do setpoint. A palavra de controle IPOS^{plus®} é conectada por um comando OR com as funções do borne, com a palavra de controle fieldbus e a palavra de controle no equipamento através da RS-485/RS-232 e do SBus.
<table>
<tr><th>Bit</th><th>Função no nível "1"</th><th>Bit</th><th>Função no nível "1"</th></tr>
<tr><td>0</td><td>Sem função</td><td>16</td><td>Reservado</td></tr>
<tr><td>1</td><td>Sem liberação</td><td>17</td><td>Came de referência</td></tr>
<tr><td>2</td><td>Horário</td><td>18</td><td>Partida de referenciamento</td></tr>
<tr><td>3</td><td>Antihorário</td><td>19</td><td>Operação livre escravo</td></tr>
<tr><td>4</td><td>n11/n21 (setpoint fixo 1)</td><td>20</td><td>Setpoint bloqueado</td></tr>
<tr><td>5</td><td>n12/n22 (setpoint fixo 2)</td><td>21</td><td>Reservado</td></tr>
<tr><td>6</td><td>Comutação do setpoint fixo</td><td>22</td><td>Ajuste ponto zero DRS</td></tr>
<tr><td>7</td><td>Comutação do parâmetro (jogo parâm. 2)</td><td>23</td><td>DRS slave start</td></tr>
<tr><td>8</td><td>Comutação da rampa (ajuste rampa 2)</td><td>24</td><td>DRS teach in</td></tr>
<tr><td>9</td><td>Potenciômetro do motor acel.</td><td>25</td><td>Reservado</td></tr>
<tr><td>10</td><td>Potenciômetro do motor desacel.</td><td>26</td><td>Reservado</td></tr>
<tr><td>11</td><td>Irregularidade externa</td><td>27</td><td>Reservado</td></tr>
<tr><td>12</td><td>Reset de irregularidade</td><td>28</td><td>Reservado</td></tr>
<tr><td>13</td><td>Controle de retenção</td><td>29</td><td>Reservado</td></tr>
<tr><td>14</td><td>Chave fim-de-curso Horária</td><td>30</td><td>Regulador bloqueado</td></tr>
<tr><td>15</td><td>Chave fim-de-curso Antihorária</td><td>31</td><td>Reservado</td></tr>
</table></td></tr>
<tr><td>485</td><td>T0_Reload / T0 RELOAD</td><td>READ e SET carregam valor para o tempo de ciclo do usuário timer 0. O tempo de ciclo pode ser especificado com H485 se for utilizado um timer do usuário (TIMER0 (H489)) com o comando SET INTERRUPT (SETINT). O valor do tempo inserido em H485 é recarregado automaticamente com este valor de tempo, cada vez que o timer 0 reduzir (H489 = 0). Faixa de valor: 0 ... 2^{31} –1 ms.</td></tr>
<tr><td>486</td><td>Reserved</td><td></td></tr>
<tr><td>487</td><td>Timer_2 / TIMER 2</td><td>Tempo para o timer 2 do usuário, READ e SET.
O timer 2 do usuário é de contagem crescente.
Faixa de valor: 0 ... 2^{31} –1 ms.</td></tr>
<tr><td>488</td><td>Timer_1 / TIMER 1</td><td>Tempo para o timer 1 do usuário, READ e SET.
O timer 1 do usuário é de contagem decrescente.
Faixa de valor: 0 ... 2^{31} –1 ms.</td></tr>
<tr><td>489</td><td>Timer_0 / TIMER 0</td><td>Tempo para o timer 0 do usuário, READ e SET.
O timer 0 do usuário é de contagem decrescente. É realizada uma rotina de interrupção quando o valor do timer alcança 0, se o comando SET INTERRUPT (SETINT) estiver sendo utilizado. O tempo de ciclo pode ser especificado com a variável T0 RELOAD (H485) se for utilizado um timer do usuário com o comando SET INTERRUPT (SETINT). Ver o capítulo „Gerenciamento do task e interrupções".
Faixa de valor: 0 ... 2^{31} –1 ms.</td></tr>
</table>

Nr.	Nome Compiler / assembler	Descrição
490	WdogTimer / WD.TIMER	**Tempo para o watchdog do usuário, READ e SET.** O timer watchdog é de contagem decrescente. O comando WATCHDOG ON (WDON) ativa o timer e determina o tempo do ciclo. Faixa de valor: 0 ... 2^{31} –1 ms.
491	SetpointPos / SETP.POS.	STOP **Posição do setpoint atual, READ.** **AVISO: Variável controlada pelo sistema! O valor não deve ser sobrescrito!** A posição do setpoint sempre tem a seguinte unidade, independente da quantidade de pulsos por volta do encoder: 4096 Inc. / volta do motor (número de pulsos por volta do encoder ≥ 512). A posição do setpoint atual representa a posição **absoluta** que é atualmente válida para controle de posição no curso do trabalho em andamento. As mudanças na posição do setpoint em intervalos de tempo resultam do perfil de deslocamento calculado com relação a rampa de posicionamento, velocidade de deslocamento, forma da rampa, etc. O valor de H491 é o mesmo de H492 uma vez o trabalho de deslocamento ter sido completado e o conversor tiver parado. Faixa de valor: -2^{31} ... 0 ... 2^{31} –1 inc.
492	TargetPos / TARGET POS	**Posição final atual, READ e SET.** A posição final sempre tem a seguinte unidade, apesar da contagem de pulso do encoder por volta: 4096 Inc. / volta do motor (número de pulsos por volta do encoder ≥ 512). Esta variável representa a posição final atual do trabalho de deslocamento em andamento. A posição é representada em H492 em condições absolutas. Exemplo: 1. Posição atual do conversor: 50000 Inc. 2. GOR NOWAIT #–8000 Inc. 3. Posição final atual: 42000 inc. Faixa de valor: -2^{31} ... 0 ... 2^{31} –1 inc. Se H492 é escrito diretamente (não utilizando o comando GO), H473, bit 19 "In position" ficará em nível 1 por até 1 ms.
493	PosWindow / POS.WINDOW	**Janela de posicionamento, READ e SET.** H494 é idêntico ao P923. A janela de posicionamento define uma faixa de distância perto da posição final (H492) do comando de deslocamento ou parada (GOx or ASTOP TARGET POSITION). Assim que o conversor tiver alcançado a janela de posicionamento, o sinal "IPOS IN POSITION" é gerado. Este sinal é mostrado através da saída digital, que pode ser ajustada com a função "IPOS IN POSITION". Este valor também pode ser visto na variável do sistema H473 bit 19. O sinal "IPOS IN POSITION" é reiniciado imediatamente quando um novo comando GO é enviado. O controle da janela de posicionamento sempre ocorre, contanto que um modo de operação com IPOS esteja ativo (P700). A precisão de posicionamento não é afetada pelo tamanho da janela de posição. Faixa de ajuste: 0 ... **50** ... 2^{15} – 1 incrementos
494	LagWindow / LAG WINDOW	**Janela de erro por atraso, READ e SET.** H494 é idêntico ao P923. A janela de erro por atraso define a diferença máxima permitida entre a posição do setpoint atual, que o gerador de rampa especifica a cada 1 ms, e a posição atual. F42 (lag error) é ativado se o valor ajustado for excedido. A reação do F42 deve ser ajustada utilizando o parâmetro P834 "Response DRS LAG ERROR". **Desativar**: Pode-se desativar o controle do erro por atraso pelo ajuste de P923 Lag error window para 0. Faixa de ajuste: 0 ... **5000** ... 2^{31} – 1 incrementos
495	LagDistance / LAG DISTAN	**Lag distance, READ.** Valor da distância de atraso atual no posicionamento (diferença entre setpoint e posição atual). Faixa de valor: 0 ... 2^{31} – 1 incrementos
496	SLS_right / SLS RIGHT	**Chave fim-de-curso horária do software, READ e SET.** H496 é idêntico ao P920. Limita a faixa de deslocamento no sentido horário. O valor é dado nas unidades de deslocamento do usuário. Faixa de ajuste: – 2^{31} ... **0 unidades do usuário** ... 2^{31} – 1 incrementos
497	SLS_left / SLS LEFT	**Chave fim-de-curso antihorária do software, READ e SET.** H497 é idêntico ao P921. Limita a faixa de deslocamento no sentido antihorário. O valor é dado nas unidades de deslocamento do usuário. Faixa de ajuste: – 2^{31} ... **0 unidades do usuário** ... 2^{31} – 1 incrementos
498	RefOffset / REF.OFFSET	**Offset de referência, READ e SET.** H498 é idêntico ao P900. O offset de referência possibilita movimentar o zero da máquina sem alterar fisicamente a posição de referência. Aplica-se a seguinte fórmula: **Zero da máquina = posição de referência + offset de referência** Durante o referenciamento, o conversor movimenta-se para a posição de referência, pára e permanece lá. O zero da máquina não é calculado até depois do referenciamento, utilizando a posição de referência e o offset de referência. O offset de referência é dado nas unidades de deslocamento do usuário. Faixa de ajuste: – 2^{31} ... **0** ... + 2^{31} – 1

<table>
<tr><th>Nr.</th><th>Nome
Compiler /
assembler</th><th>Descrição</th></tr>
<tr><td>499</td><td>SetpPosBus /
SP.POS.BUS</td><td>Rede de posição do setpoint, READ.
Contém a posição do setpoint que é enviado através dos dados de processo fieldbus. A posição do setpoint somente é aceita se 'POSITION LO' e 'POSITION HI' estiverem programados na descrição de dados PO (grupo de parâmetro P87_).</td></tr>
<tr><td>500</td><td>TpPos2_VE /
TP.POS2VE</td><td>Utilizado somente no MOVIDRIVE® B; reservado no MOVIDRIVE® A.
O valor do encoder virtual H376 é armazenado no H599 se a entrada DI03 tiver sido ativada (ver também _TouchProbe() / TOUCHP).</td></tr>
<tr><td>501</td><td>TpPos1_VE /
TP.POS1VE</td><td>Utilizado somente no MOVIDRIVE® B; reservado no MOVIDRIVE® A.
O valor do encoder virtual H376 é armazenado no H501 se a entrada DI02 tiver sido ativada (ver também _TouchProbe() / TOUCHP).</td></tr>
<tr><td>502</td><td>TpPos2_Abs /
TP.POS2ABS</td><td rowspan="6">As posições touch probe são armazenadas nas seguintes variáveis:

Encoder | Posição do Encoder | Touch probe 1 DI02 | Touch probe 2 DI03
Encoder do motor (X15) | H511 ACTPOS.MOT | H507 TP.POS1MOT | H505 TP.POS2MOT
Encoder externo (X14) | H510 ACTPOS.EXT | H506 TP.POS1EXT | H504 TP.POS2EXT
Encoder absoluto (X62) | H509 ACTPOS.ABS | H503 TP.POS1ABS | H502 TP.POS2ABS</td></tr>
<tr><td>503</td><td>TpPos1_Abs /
TP.POS1ABS</td></tr>
<tr><td>504</td><td>TpPos2_Ext /
TP.POS2EXT</td></tr>
<tr><td>505</td><td>TpPos2_Mot /
TP.POS2MOT</td></tr>
<tr><td>506</td><td>TpPos1_Ext /
TP.POS1EXT</td></tr>
<tr><td>507</td><td>TpPos1_Mot /
TP.POS1MOT</td></tr>
<tr><td>508</td><td colspan="2">Reserved</td></tr>
<tr><td>509</td><td>ActPos_Abs /
ACTPOS ABS</td><td>Posição atual do encoder absoluto DIP (SSI), READ.
AVISO: Variável controlada do sistema! O valor não deve ser sobrescrito!
Esta posição atual é determinada através dos sinais ativos no conector X62 (opcional DIP11A).
Unidade: Incrementos dependendo do número de pulsos por volta do encoder.</td></tr>
<tr><td>510</td><td>ActPos_Ext /
ACTPOS EXT</td><td>Posição atual do encoder externo, READ
AVISO: Variável controlada do sistema! O valor não deve ser sobrescrito!
A posição atual é determinada através dos sinais ativos no conector X14. A descoberta da posição somente é realizada se o conector X14 for utilizado como entrada do encoder.
Unidade: Incrementos dependendo do número de pulsos por volta do encoder.</td></tr>
<tr><td>511</td><td>ActPos_Mot /
ACTPOS MOT</td><td>Posição atual do encoder do motor, READ.
AVISO: Variável controlada do sistema! O valor não deve ser sobrescrito!
A posição atual sempre tem a seguinte unidade, apesar da contagem de pulso do encoder por volta: 4096 incrementos por volta do motor (número de pulsos por volta do encoder 512 inc., exceto: MQx com NV26 tem 24 Incrementos por volta do motor).</td></tr>
</table>

Além disso, as variáveis a seguir são funções atribuídas ou reservadas no MOVIDRIVE® B:

<table>
<tr><th>No.</th><th>Nome Compiler / assembler</th><th>Descrição</th></tr>
<tr><td>512
...
519</td><td>Reserved</td><td></td></tr>
<tr><td>520</td><td>InpLevelB / INPUTLVLB MOVIDRIVE® B (MOVIDRIVE® A: H483)</td><td>Nível de sinal das entradas digitais, somente LEITURA.
<table>
<tr><th>Bit</th><th>Nome IPOS</th><th>DIO (+DIO)</th><th>DIO + DIP</th><th>DIO + - fieldbus</th><th>DIP</th><th>DIP + fieldbus</th><th>Fieldbus P870 = palavra de controle 2</th></tr>
<tr><td>0
1
2
3
4
5
6
7</td><td>DI00
DI01
DI02
DI03
DI04
DI05
DI06
DI07</td><td colspan="6">Depende da unidade básica, por ex. X13:1
X13:2
X13:3
X13:4
X13:5
X13:6
X16:1
X16:2</td></tr>
<tr><td>8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23</td><td>DI10
DI11
DI12
DI13
DI14
DI15
DI16
DI17</td><td>X22:1
X22:2
X22:3
X22:4
X22:5
X22:6
X22:7
X22:8

(X22:1)
...
(X22:8)</td><td>X22:1
X22:2
X22:3
X22:4
X22:5
X22:6
X22:7
X22:8
X60:1
X60:2
X60:3
X60:4
X60:5
X60:6
X60:7
X60:8</td><td>X22:1
X22:2
X22:3
X22:4
X22:5
X22:6
X22:7
X22:8</td><td>X60:1
X60:2
X60:3
X60:4
X60:5
X60:6
X60:7
X60:8</td><td>X60:1
X60:2
X60:3
X60:4
X60:5
X60:6
X60:7
X60:8</td><td>Bit 8
Bit 9
Bit 10
Bit 11
Bit 12
Bit 13
Bit 14
Bit 15</td></tr>
</table></td></tr>
<tr><td>521</td><td>OutpLevelB / OUTPUTLVLB MOVIDRIVE® B (MOVIDRIVE® A: H482)</td><td>Nível de sinal das saídas digitais, somente LEITURA.
<table>
<tr><th>Bit</th><th>Nome IPOS</th><th>DIO (+DIO)</th><th>DIO + DIP</th><th>DIO + - fieldbus</th><th>DIP</th><th>DIP + fieldbus</th><th>Fieldbus P873 = palavra de estado 2</th></tr>
<tr><td>0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21</td><td>DB00
DO01
DO02
DO03
DO04
DO05
DO10
DO11
DO12
DO13
DO14
DO15
DO16
DO17</td><td>DB00
DO01
DO02
DO03
DO04
DO05
X23:1
X23:2
X23:3
X23:4
X23:5
X23:6
X23:7
X23:8

(X23:1)
...
(X23:8)</td><td>DB00
DO01
DO02
DO03
DO04
DO05
X23:1
X23:2
X23:3
X23:4
X23:5
X23:6
X23:7
X23:8
X61:1
X61:2
X61:3
X61:4
X61:5
X61:6
X61:7
X61:8</td><td>DB00
DO01
DO02
DO03
DO04
DO05
X23:1
X23:2
X23:3
X23:4
X23:5
X23:6
X23:7
X23:8</td><td>DB00
DO01
DO02
DO03
DO04
DO05
X61:1
X61:2
X61:3
X61:4
X61:5
X61:6
X61:7
X61:8</td><td>DB00
DO01
DO02
DO03
DO04
DO05
X61:1
X61:2
X61:3
X61:4
X61:5
X61:6
X61:7
X61:8</td><td>DB00
DO01
DO02
DO03
DO04
DO05
Bit 8
Bit 9
Bit 10
Bit 11
Bit 12
Bit 13
Bit 14
Bit 15</td></tr>
</table></td></tr>
</table>

No.	Nome Compiler / assembler	Descrição
522	RecStatS1 / SBUS1REC	**Palavra de estado para receber os objetos de dados SCOM (double words) através da rede de comunicação** Um bit recebido é reservado para cada objeto de dados configurado utilizando _SBusCommDef / SCOM. Ao primeiro objeto recebido inicializado no programa IPOS é atribuído bit 0, ao segundo é atribuído bit 1 e assim por diante. Quando o MOVIDRIVE® recebe uma mensagem de um objeto inicializado, o bit correspondente é ajustado. O bit somente pode ser reiniciado no programa do usuário. As mensagens da ocorrência da máquina podem ser enviadas e recebidas através do SBUS se uma interrupção da variável tiver sido ajustada para o bit correspondente em H522. O reinício do bit deve constituir o último comando na rotina de interrupção. Ao projetar uma imagem de processo, o usuário deve assegurar que nenhum efeito lateral seja causado quando o mesmo objeto for recebido durante o processamento (recibo cíclico de um objeto). Para reiniciar o bit, utilizar o comando BITCLEAR para que os bits recebidos em outras transferências não sejam perdidos.
523	RecStatS2 / SBUS2REC	**Somente com CAN bus via DFC11B: Palavra de estado para o recibo de objetos de dados SCOM (double words) via CAN bus** Um bit recebido é reservado para cada objeto de dados configurado utilizando _SBusCommDef / SCOM. O primeiro objeto recebido inicializado no programa IPOS é atribuído bit 0, o segundo é atribuído bit 1 e assim por diante. Quando o MOVIDRIVE® recebe uma mensagem de um objeto inicializado, o bit correspondente é ajustado. O bit somente pode ser reiniciado no programa do usuário. As mensagens da ocorrência da máquina podem ser enviadas e recebidas através do SBUS se uma interrupção da variável tiver sido ajustada para o bit correspondente em H523. O reinício do bit deve constituir o último comando na rotina de interrupção. Ao projetar uma imagem de processo, o usuário deve assegurar que nenhum efeito lateral seja causado quando o mesmo objeto é recebido durante o processamento (recibo cíclico de um objeto). Para reiniciar o bit, utilizar o comando BITCLEAR para que os bits recebidos em outras transferências não sejam perdidos.
524	IPOS_Setp / IPOS_SETP	**Setpoint IPOS, variável de correção do controlador PID quando H540 = 1.** Quando H540 = 0 ou 2 o setpoint também pode ser escrito diretamente a partir do programa do usuário. H524 pode ser utilizado como setpoint de torque ou setpoint de rotação quando P100 Fonte do setpoint = IPOS e P700 Modo de operação 1 = xxx&M-control, CFC ou SERVO. 1 incremento em H524 então corresponde a 0.21 rpm setpoint de rotação ou 0.01 % I_N setpoint de torque
525 ... 529	Reserved	
530	VarIntReq / VARINTREQ	Caso o bit solicitado correspondente de uma interrupção da variável seja ajustado, uma interrupção da variável é ativada independente da condição de interrupção atual. A interrupção da variável relacionada deve ser ativada antes. Bit 0: Pedido para interrupção da variável 0 Bit 1: Pedido para interrupção da variável 1 Bit 2: Pedido para interrupção da variável 2 Bit 3: Pedido para interrupção da variável 3

No.	Nome Compiler / assembler	Descrição
As variáveis 540 ... 559 devem estar disponíveis nos equipamentos padrão a partir do 2º semestre de 2004.		
540	PID_Mode / PID.MODE	Modo de operação do controlador PID, H540 e P260 são idênticos. 0 = Controlador desativado (default) 1 = Controle ativo 2 = Reação da fase (sistema de controle aberto)
541	PID_K_p / PID.KP	Controlador PID: Fator do componente proporcional, H541 e P263 são idênticos, 3 casas decimais; 0 <= K_P <> 32000 (= 32,000); padrão: 1000 (= 1,0)
542	PID_Outp_P / PID.OUTPP	Controlador PID: Valor atual do componente P do controlador
543	PID_Outp_I / PID.OUTPI	Controlador PID: Valor atual do componente I do controlador O valor para o componente I é na palavra mais significativa, os componentes internos para as casas decimais são na palavra menos significativa, por exemplo, H543 = 0x30000 ⇒ Componente I = 3.
544	PID_Outp_D / PID.OUTPD	Controlador PID: Valor atual do componente D do controlador
545	PID_Feedf / PID.FEEDF	Valor pré-definido do controlador PID; H545 e P266 são idênticos –32000 <= Pré-ajuste <= 32000; padrão: 0
546	PID_Command / PIDCOMMAND	Controlador PID: Setpoint, H546 e P271 são idênticos Quando P270 = 0 (= "Parameter"), P271 / H546 contém o setpoint do processo necessário –32000 <= Setpoint <= 32000 (para controle de rotação, 1 incremento corresponde a 0.2/min); padrão: 0
547	PID_CmdAdr / PID.CMDADR	Controlador PID: Endereço do setpoint, H547 e P272 são idênticos Quando P270 = 1 (= "IPOS variable"), P272 / H547 contém o endereço da variável IPOS com o setpoint; padrão: 0
548	PID_CmdScale / PID.CMDSCA	Controlador PID: Fator para escala do setpoint, H548 e P274 possuem pesos idênticos com 3 casas decimais –32000 (–32,000) <= $K_{Setpoint}$ <= 32000 (32,000); padrão: 1000 (1,0)
549	PID_ActAdr / PID.ACTADR	Controlador PID: Endereço do valor atual, H549 e P276 são idênticos Quando P275 = "IPOS variable", P276 / H549 contém o endereço da variável IPOS; padrão: 0
550	PID_ActScale / PID.ACTSCA	Controlador PID. Fator de escala do valor atual filtrado, H550 e P277 possuem pesos idênticos com 3 casas decimais –32,000 <= $K_{Valor\ atual}$ <= 32,000; padrão: 1000 (1,0)
551	PID_ActNorm / PID.ACTNOR	Controlador PID: Valor atual filtrado e escalonado, valor de diagnóstico
552	PID_ActOffset / PID.ACTOFF	Controlador PID: Inteiro, offset permanente do valor atual, H552 e P278 são idênticos –32000 (–32000) <= Offset <= 32000 (32000); padrão: 0
553	PID_ActMin / PID.ACTMIN	Controlador PID: Valor mínimo para valor atual após suavização, escala e offset, H553 e P280 são idênticos. –32000 (–32000) <= $x_{e,min}$ <= 32000 (32000); padrão: 0
554	PID_ActMax / PID.ACTMAX	Controlador PID: Valor máximo para valor atual após suavização, escala e offset, H554 e P281 são idênticos. –32000 (–32000) <= $x_{e,max}$ <= 32000 (32000); padrão: 10000 (10,0)
555	PID_LimitMin / PID.LMTMIN	Controlador PID: Valor mínimo de saída, H555 e P282 são idênticos –32000 (–32000) <= $x_{Controller,min}$ <= 32000 (32000); padrão: –1000 (–1,0)
556	PID_Limit_Max / PID.LMTMAX	Controlador PID: Valor máximo de saída, H556 e P283 são idênticos –32000 (–32000) <= $x_{Controller,max}$ <= 32000 (32000); padrão: 10000 (1,0)
557	PID_SetpMin / PID.SETMIN	Controlador PID: Valor mínimo de saída para variável de correção, H557 e P284 são idênticos –32000 (–32000) <= $x_{a,min}$ <= 32000 (32000); padrão: 0
558	PID_SetpMax / PID.SETMAX	Controlador PID: Valor máximo de saída para variável de correção, H558 e P285 são idênticos –32000 (–32000) <= $x_{a,max}$ <= 32000 (32000); padrão: 7500 (7,5)
559	PID_Status / PID.STATUS	Palavra de estado do controlador PID Bit 0 = Total do valor atual e offset excede limite xe_{min} Bit 1 = Total do valor atual e offset excede limite xe_{max} Bit 2 = Valor do componente P do controlador é limitado Bit 3 = Componente I do controlador é desativado Bit 4 = Valor do componente I do controlador é limitado Bit 5 = Valor do componente D do controlador é limitado Bit 6 = Variável de correção do controlador PID é limitada Bit 7 = Total da variável de correção do controlador PID e freqüência fixa é limitado
560	Reserved	

4 Gerenciamento do Task e Interrupções

4.1 Introdução

O IPOS$^{plus®}$ pode processar vários subprogramas ao mesmo tempo. Um subprograma corresponde a um task. As funções a seguir podem desarmar interrupções para task 1:

- Overflow do Timer0
- Erro de sistema / erro do equipamento
- Touch probe DI02

O MOVIDRIVE® B pode interromper task 2 e task 3 com 4 interrupções adicionais, que são ativadas pela comparação com um valor variável.

O MOVIDRIVE® A pode realizar 2 subprogramas – task1 e task 2 – operando independente um do outro.

O MOVIDRIVE® B pode realizar 3 subprogramas – task1, task 2 e task 3 – operando independente um do outro.

O MOVIDRIVE® B pode ser operado da mesma maneira que o MOVIDRIVE® A.

As variáveis a seguir são globais. Todos os tasks e interrupções utilizam as mesmas variáveis:

- H0 - H511 para MOVIDRIVE® A
- H0 - H1023 para MOVIDRIVE® B

Não há variáveis locais que são declaradas somente em uma task ou em uma função. Para um resumo da área reservada para as variáveis do sistema, ver a seção "Variáveis IPOS / Resumo das variáveis do sistema."

Pode-se inserir os comandos na janela do programa. Deve-se criar todas as seções do programa utilizando a mesma linguagem: Assembler ou compiler.

4.2 Gerenciamento do task para MOVIDRIVE® A e B

O Task 1 é o programa principal. Pode-se iniciar o task 1 utilizando o símbolo de raio na barra de ferramenta ou utilizando o controle manual (P931). Inicializar o task 2 e o task 3 utilizando os comandos do programa. O Task 3 é disponível somente com o MOVIDRIVE® B. A interrupção é inicializada utilizando os comandos do programa e ativada utilizando um evento de interrupção. Caso páre o task 1 utilizando o ícone STOP na barra de ferramenta, este pára o processamento IPOS inteiro. O controle manual DBG pode ser utilizado para parar todos os tasks pelo ajuste do parâmetro P931 para STOP. Quando P931 = STOP, somente o task 1 é parado. Após reiniciar, o programa começa novamente no primeiro comando.

Barra de ferramenta para MOVIDRIVE® A

[1] [2] [3] [4]

[1] = Estado do task 1: START = em funcionamento
[2] = Estado do task 2: PSTOP = parado
[3] = Ícone raio para iniciar task 1
[4] = Ícone STOP para parar o processamento IPOS inteiro

Barra de ferramenta para MOVIDRIVE® B

[1] [2] [3] [4] [5]

[1] = Estado do task 1: PSTOP = parado
[2] = Estado do task 2: PSTOP = parado
[3] = Estado do task 3: PSTOP = parado
[4] = Ícone raio para iniciar task 1
[5] = Ícone STOP para parar o processamento IPOS inteiro

O IPOS$^{plus®}$ processa um task ciclicamente. Um task inicia novamente com o primeiro comando, uma vez o último comando ter sido completado. Caso queira execu-tar uma vez uma rotina de inicialização no task 1, pode-se excluir esta seção do programa utilizando uma rotina contínua.

```
main ()
{
  // Program code only initialization
  while(1)
  {
    // Cyclic program code
  }
}
```

A tabela a seguir resume as funções e propriedades dos tasks e interrupções.

	Task 1	Task 2	Task 3 (somente MOVIDRIVE® B)	Interrupção "Task1"	Interrupção variável (somente MOVIDRIVE® B)
Início MOVIDRIVE® A	Ícone raio ou P931 no controle manual	_SetTask2(ST2_START, Task2Name); or TASK2 START Mxx	Não disponível	Com o evento de interrupção definido	Não disponível
Início MOVIDRIVE® B		_SetTask(ST2_START, Task2Name); or TASK TASK2, START Mxx	_SetTask(ST3_START, Task3Name); ou TASK TASK3, START Mxx		Com o evento de interrupção definido
Parada MOVIDRIVE® A	Ícone STOP ou P931 no controle manual	STOP icon or _SetTask2(ST2_STOP, Task2Name); or TASK2 STOP Mxx	Não disponível	Como task 1	Não disponível
Parada MOVIDRIVE® B		STOP icon or _SetTask(ST2_STOP, Task2Name); or TASK TASK2, STOP Mxx	Ícone STOP ou _SetTask(ST3_STOP, Task3Name); ou TASK TASK3, STOP Mxx		Como task 1 ou como task 2 ou 3 atribuída
Interrupção MOVIDRIVE® A	Com interrupção "Task1"	Não pode ser interrompido	Não disponível	Com outra interrupção task 1 com prioridade mais alta	Não disponível
Interrupção MOVIDRIVE® B	Com interrupção "Task1"	Com interrupção variável	Com interrupção variável	Com outra interrupção task 1 com prioridade mais alta	Com interrupção variável no mesmo task com prioridade mais alta
Debug com ponto de inflexão e fase simples	Sim	Não[1)]	Não[1)]	Sim	Não[1)]
Tempo de processamento do comando - MOVIDRIVE® A	1 comando assembler / ms	2 comandos assembler / ms	Não disponível	Comando Assembler / ms	Não disponível
Tempo de processamento do comando -MOVIDRIVE® B	1 ... 10 comandos assembler / ms, ajuste de fábrica: 1 comando / ms	2 ... 11 comandos assembler / ms, ajuste de fábrica: 2 comandos / ms	Pelo menos 1 comando por ms; comandos adicionais são processados dependendo da utilização do processador	Como task 1	Como task para qual a interrupção é atribuída (task 2 ou 3)

1) Comando de cópia do task 1 para debugging.

4.3 Tarefas para MOVIDRIVE® A

Além da informação geral na seção "Gerenciamento do Task para MOVIDRIVE® A e B," esta seção contém informação específica de implementação para MOVIDRIVE® A:

A seqüência de movimento com os comandos de posicionamento é programada no task 1.

Programar as seguintes funções no task 2:

- Processos de tempo crítico, rápido
- Cálculos
- Monitoração para valores do sistema
- Comunicação com o painel de operação SEW
- Copiar variáveis ciclicamente para as variáveis osciloscópio H474, H475
- Formatar os dados de processo fieldbus / SBus com um controle mestre ou outro MOVIDRIVE®

Deste modo, o IPOS$^{plus®}$ também realiza estas funções quando a rotina de interrupção está ativa no task 1.

4.4 Tarefas para MOVIDRIVE® B

Além da informação geral na seção "Gerenciamento do Task para MOVIDRIVE® A e B," esta seção contém informação específica de implementação para MOVIDRIVE® B:

Tempo de processamento task 1 / task 2

Ajuste de fábrica para o tempo de processamento dos tasks:

- Task 1: 1 comando / ms (P938 = 0)
- Task 2: 2 comandos / ms

Pode-se acelerar o tempo de processamento dos dois tasks juntos até 9 comandos adicionais por ms. Pode-se atribuir os comandos adicionais para task 1 via parâmetro P938 (index 8888) e para task 2 via parâmetro P939 (index 8962). Isto quer dizer que pode ser realizado no máximo 1 + 9 = 10 comandos / ms no task 1 e no máximo 2 + 9 = 11 comandos / ms no task 2.
Se dividir o número máximo de nove comandos adicionais / ms entre tasks 1 e 2, terá as seguintes combinações:

Task 1		Task 2	
P938	**Comandos / ms**	**P939**	**Comandos / ms**
0	1	9	11
1	2	8	10
2	3	7	9
3	4	6	8
4	5	5	7
5	6	4	6
6	7	3	5
7	8	2	4
8	9	1	3
9	10	0	2

Exemplo: P938 = 2, P939 = 3 Task 1 processa 3 comandos / ms, task 2 processa 5 comandos / ms.

Task3

O Task 3 é disponível nos equipamentos da série B. O Task 3 processa pelo menos 1 comando / ms. Dependendo da configuração do equipamento e do ajuste de P938 / P939, o task 3 realizará comandos adicionais. 20 ... 40 comandos / ms é o número típico de comandos para o task 3. As exigências de recurso absoluto para processamento do comando é aprox. 20 ... 40 % inferior no task 3 comparado com o task 1 ou task 2. Os componentes da aplicação, para os quais o tempo de funcionamento garantido das linhas de programa individuais não é importante, são processados mais rápido no task 3.

Informação para a implementação

Ao contrário do MOVIDRIVE® A, programar a seqüência de movimento no task 1 ou task 3.

Programar as seguintes funções no task 2 ou task 3:

- Processos de tempo crítico, rápido
- Cálculos
- Monitoração para valores do sistema
- Comunicação com o painel de operação SEW
- Copiar variáveis ciclicamente para as variáveis osciloscópio H474, H475
- Formatar os dados de processo fieldbus / SBus com um controle mestre ou outro MOVIDRIVE®

Deste modo, o IPOS$^{plus®}$ também realiza estas funções quando a rotina de interrupção está ativa no task 1.

Observar que ao contrário do task 3, o tempo de processamento por comando no task 2 é determinante.

Exemplo

O MOVIDRIVE® B posiciona um transportador. Um CLP controla o MOVIDRIVE® através de uma rede fieldbus. Mudar os parâmetros individuais diretamente no MOVIDRIVE® utilizando o controle manual SEW.

Solução proposta:

Task 1: Programação da seqüência de movimento

Task 2: Comunicação IHM com o painel de operação

Task 3: Comunicação Fieldbus com o CLP

É importante distribuir os comandos adicionais corretamente, dependendo da aplicação:

- **Programas com interrupção orientada:** Quando um programa do usuário possui interrupção orientada e esta deve ser processada rapidamente, o task 1 deve ser ajustado a uma alta prioridade de cálculo utilizando o parâmetro P938.
- **Programas de operação otimizada:** Se, por exemplo, os dados de processo forem convertidos em IPOS, este deve ser completado o mais rápido possível. O Task 3 pode ser utilizado para processar rotinas de conversão o mais rápido possível. Neste caso, o task 1 e o task 2 devem ser destinados ao menor número de comandos adicionais /ms possível. Isto garante o mais rápido desempenho total da aplicação se os tasks 1 e 2 funcionam com velocidade mínima.

4.5 Interrupções

Uma interrupção – ativada por um evento – interrompe o processamento do task que ela é destinada. A rotina inteira de interrupção funciona completamente uma vez, contanto que ela não seja interrompida com uma prioridade maior do mesmo task.

Uma interrupção que é ativada por _SetInterrupt() ou SETINT pode ser ativada por um sobrefluxo timer0, uma irregularidade do sistema/equipamento ou touch probe DI02. Isto interrompe o task 1.

No MOVIDRIVE® B, podem ser ativadas até 4 interrupções variáveis adicionais utilizando _SetVarInterrupt() ou VARINT. Elas interrompem o task 2 ou task 3 como desejado.

O tempo de reação para interrupção do task 1 (irregularidade do equipamento, DI02 touch probe ou T0_overflow) depende do número de interrupções ativadas (1 interrupção <= 1ms, 2 interrupções <= 2ms, 3 interrupções <= 3ms). O tempo de reação para interrupções variáveis depende do número de interrupções ativadas <= 1ms.

Se uma interrupção é ativada durante um comando de espera, o tempo de espera do comando continua a funcionar. Uma vez o programa ter voltado para o task, esperar somente o tempo restante antes de continuar.

Exemplo

Um comando WAIT 1000 ms no task 1 é interrompido após 500 ms. Se o processamento do task 1 é recomeçado após 175 ms, a operação restante é 325 ms.

4.6 Interrupções para MOVIDRIVE® A e B

Podem ser utilizadas as seguintes interrupções no MOVIDRIVE® A:

- Interrupção overflow do Timer0 (H489) Prioridade = 1 (prioridade mais baixa)
- Interrupção Touch probe DI02 Prioridade = 2
- Interrupção por erro Prioridade = 3 (prioridade mais alta)

O Task 1 pode ser interrompido diversas vez. Na teoria, um timer0, um touch probe e um error interrupt podem estar ativos ao mesmo tempo. Uma interrupção atribuída a uma prioridade mais alta pode interromper o processamento de outra interrupção. O argumento DISABLE desativa todas as interrupções. (Ver _SetInterrupt or SETINT)

Ativar a interrupção

Compiler: `_SetInterrupt(event ,myfunction );`

Ativa uma interrupção. Quando ocorre o event, a função `myfunction` é realizada em vez do task 1.

Assembler: `SETINT event, Mxx`

Ativa uma interrupção. Quando ocorre o event, os comandos da label Mxx são realizados em vez do task 1.

Interrupção Error

A rotina inteira de interrupção funciona completamente uma vez, se ocorrer uma irregularidade. Após uma volta completa, é processado um comando assembler do task 1 antes do sistema verificar se a irregularidade ainda ocorre. Se a irregularidade ainda estiver presente, a rotina de interrupção é processada novamente. Para permanecer na rotina de interrupção até a irregularidade não estar mais presente, a rotina deve conter um circuito que realize esta condição.

Dependendo da reação de irregularidade ajustada no grupo de parâmetro 83x ou com o comando _FaultReaction ou SETFR, o programa age conforme segue:

- Nenhuma interrupção é ativada para uma irregularidade específica se a reação desta irregularidade for ajustada para NO RESPONSE.
- Se a reação de uma irregularidade xy for ajustada para "... and warning", o task 1 continua do mesmo lugar, uma vez a irregularidade ter sido reativada.
- Se a reação de uma irregularidade xy é ajustada para "... and fault", o IPOS é reiniciado, uma vez a irregularidade ter sido reativada e as variáveis são reinicializadas com os valores da EEPROM. Observe: Neste caso, pode-se armazenar valores importantes da variável protegidos contra falha de potência com o comando MEM ou MOVILINK antes da reativação da irregularidade. Fazendo assim, observar que o número de acessos escritos admissíveis não deve ser excedido (Ver MEM ou _Memorize()).

Exemplo

```
fnErrorInterrupt()
{
  H2++;
  while( !(StatusWord & 0b10))
  {
    //Only leave the while-loop when drive is fault-free
    H1++;
  }
}

main()
{
  _SetInterrupt(SI_ERROR, fnErrorInterrupt);
  while(1)
  {
    H0++;
  }
}
```

H0 é incrementado contanto que o conversor esteja funcionando corretamente. Se ocorrer uma irregularidade, H2 é aumentado por um e H1 é incrementado até o conversor funcionar corretamente. Dependendo da reação de irregularidade ajustada, o sistema continua o processamento com os valores de corrente no task 1, ou o IPOS é reiniciado utilizando os valores da EEPROM.

Interrupção Touch probe DI02

A rotina inteira de interrupção funciona completamente uma vez, se a touch probe tiver sido liberada com o comando _TouchProbe (condição) ou TOUCHP e a condição de flanco seja realizada. Então, o processamento para task 1 continua. A rotina de interrupção somente é acessada uma segunda vez quando a touch probe é novamente liberada e a condição é realizada. (Ver também o comando _TouchProbe ou TOUCHP).

Exemplo

```
fnTouchInterrupt()
{
  H0++;
}
/*============================================
Main function (IPOS initial function)
============================================*/
main()
{
  _SetInterrupt( SI_TOUCHP1,fnTouchInterrupt); //Activate interrupt routine
  TouchProbe( TP_EN1 );
  while (1) { H1 = H1 +1; }
}
```

H0 é aumentado uma vez por 1.

Se o comando touch probe também é acessado no momento da rotina ou no fnTouch-Interrupt, a interrupção responde para todas as mudanças no sinal do flanco em DI02.

As aplicações típicas para a touch probe são: Posicionamento relativo para despaletizadora ou registro de controle de rotina para processos com um fluxo contínuo de material.

Interrupção Timer0

A rotina inteira de interrupção funciona completamente uma vez, quando o timer tiver ocorrido (=0). Após uma volta completa, é processado um comando assembler do task 1 antes do sistema verificar se o tempo = 0. Uma vez a condição ser realizada, o processamento volta para a rotina de interrupção.

O tempo de ciclo pode ser ajustado na variável H485 T0_Reload para ativar a interrupção do timer0 em intervalos iguais. Este tempo de ciclo é utilizado para recarregar o timer0 automaticamente, quando isto ocorre na rotina de interrupção.

Depois que o timer0 retroceder a contagem, a condição de interrupção é realizada permanentemente se T0_Reload = 0 e o valor do timer não é mudado na rotina de interrupção.

Isto resulta nas seguintes opções:

- Se uma seção do programa deve ser processada em intervalos iguais com a interrupção timer0, o timer0 deve ser recarregado, por exemplo, com T0_Reload.
- Se uma seção do programa funcionar completamente uma vez com a interrupção timer0 em um tempo definido após o IPOS ter sido iniciado, o timer0 deve ser ajustado para –1 na interrupção.

Exemplo

```
fnTimerInterrupt()
{
  H0 = H0 +1;
  T0_Reload = 10000; //Reload timer 0 automatically with 10s
  _SetInterrupt( SI_TIMER0,fnTimerInterrupt);  // Activate interrupt
}
main()
{
  while (1) {H1 = H1 +1;}
}
```

H0 é aumentado por 1 a cada 10 s.

4.7 Interrupções variáveis com MOVIDRIVE® B

Todas as interrupções no MOVIDRIVE® B são as mesmas utilizadas no MOVIDRIVE® A (ver a seção "Gerenciamento da tarefa para MOVIDRIVE® A e B), mais 4 interrupções variáveis adicionais.

As interrupções para valores variáveis específicos podem, por exemplo, reagir a

- Um valor de quantidade
- Todos os timers 0, 1 e 2
- O fato que uma posição do próprio eixo ou de um eixo diferente tenha sido alcançada
- Uma mudança em um sinal I/O
- Um certo estado do conversor (H473)
- Novo dado a ser recebido ou enviado pelo SBus

Acessando a interrupção variável

Compiler: _SetVarInterrupt(pData ,myfunction);

Ativa uma interrupção variável com a estrutura de dados igual da variável pData, que roda a função myfunction, quando ocorre a interrupção.

Assembler: VARINT Hxx, Mxx

Ativa uma interrupção variável com a estrutura de dados igual da variável Hxx, que executa os comandos como da label Mxx, quando ocorre a interrupção.

A seguir, as propriedades e funções da interrupção variável podem ser definidas na estrutura de dados (ver também o comando _SetVarInterrupt ou VARINT):

- A tarefa a ser interrompida – task 2 ou task 3
- Número seqüencial da interrupção (0... 3)
- Variável de referência e valor comparados entre si
- Tipo da comparação matemática (==, <, pulso...)
- Tempo de processamento: contanto que a condição seja realizada ou cada vez que a condição seja realizada (ativado por pulso)
- Prioridade da interrupção
- Valor da variável de referência utilizado para disparar a interrupção

A estrutura de dados do comando é descrita na função do sistema (compiler – funções / assembler – comandos).

O comportamento da interrupção pode ser adaptado dinamicamente durante a operação, por

- mudança da estrutura de dados e então acessando o comando novamente (necessário se, por exemplo, o valor CompVar utilizado para a comparação mudar) ou
- acessando o comando com uma estrutura de dados diferente mas com o mesmo valor na variável H+1 (IntNum).

Acesso IPOS ao controle interno da interrupção

A informação, como se uma interrupção variável tivesse sido pedida, pode ser vista no programa IPOS$^{plus®}$ na variável uVarEventRequest (H530 bit 0 a 3). Estes bits "Request" também podem ser escritos no programa IPOS$^{plus®}$.

uVarEventRequest H530.0	Pedido para interrupção variável 0
uVarEventRequest H530.1	Pedido para interrupção variável 1
uVarEventRequest H530.2	Pedido para interrupção variável 2
uVarEventRequest H530.3	Pedido para interrupção variável 3

Deste modo, por exemplo, o bit request pode ser ajustado para teste durante a colocação em operação inicial apesar da condição de interrupção atual e, uma interrupção variável pode ser ativada (contanto que a interrupção variável correspondente tenha sido ativada antes).

Uma interrupção variável pode ser utilizada para configurar o processamento do programa controlado pelo tempo, por exemplo, o cálculo cíclico de aceleração da rotação.

Além disso, quando uma interrupção variável de alta prioridade estiver sendo processada, pode-se apagar uma pendência, interrupção de prioridade inferior apagando o bit request correspondente.

Exemplo

O eixo para transporte de uma máquina enchedora deve mover-se para uma unidade de medição utilizando DO01 quando passar da 5ª posição na máquina. A saída deve ser desativada 200 ms depois, independente da rotação e posição do eixo.

Exemplo solucionado no compiler

Ajustes de parâmetros desejados

P620 = saída IPOS, P960 = por exemplo SHORT

```
/*==========================================
IPOS source file
==========================================*/
#include <constb.h>
#include <iob.h>

// Necessary parameter settings:
// P620 = IPOS output, P960 = for example SHORT

VARINT hOPENvalve, hCLOSEvalve;

fnTask3()
{ //Task 3 is only needed to activate VarInt.
  H1 = H1;
  //Dummy command
}

fnOPENvalve()
{ //Switch on metering unit
  Timer_2 = 0;
  //Reset timer 2 to 0
  hCLOSEvalve.Mode = 2;
  //Activate stop-IRQ
  _SetVarInterrupt( hCLOSEvalve,fnCLOSEvalve );
  _BitSet( StdOutpIPOS, 1);
  //Set DO01

}
fnCLOSEvalve()
{ //Switch off metering unit
  _BitClear( StdOutpIPOS, 1);
  //Delete DO01
  hCLOSEvalve.Mode = 0;
  //Deactivate stop-IRQ
  _SetVarInterrupt( hCLOSEvalve,fnCLOSEvalve );
}

/*==============================================
Main function (IPOS initial function)
==============================================*/
main()
{ //Initialization section
  hOPENvalve.Control = 2;        //Interrupt task3
  hOPENvalve.IntNum =0;
  //Sequential no.
  hOPENvalve.pSrcVar = numof ( ModActPos );   //Modulo motor encoder
  hOPENvalve.CompVar = 910;

  // 5° on the machine = 5° x 910/65536
  hOPENvalve.Mode = 12;          // Once when >= 5°
  hOPENvalve.Priority = 6;       // Medium priority

  hCLOSEvalve.Control = 2;
  //Interrupt task3
  hCLOSEvalve.IntNum = 1; //Sequential no.
  hCLOSEvalve.pSrcVar = numof ( Timer_2 ) ;    //Timer 2
  hCLOSEvalve.CompVar = 200;  //Deactivate after 200 ms
  hCLOSEvalve.Mode = 0;            //Deactivate timer-IR first
  hCLOSEvalve.Priority = 7;

  // Activate interrupt routine and task3
  _SetTask(ST3_START, fnTask3);
  _SetVarInterrupt( hOPENvalve,fnOPENvalve );
  /*-------------------------------------
  Main program loop
  -------------------------------------*/
  while(1)
  { //Cyclical program section ...
  } //End while (1)
} //End main()
```

Exemplo solucionado no assembler

Ajustes de parâmetros necessários:

P620 = saída IPOS, P960 = por exemplo SHORT

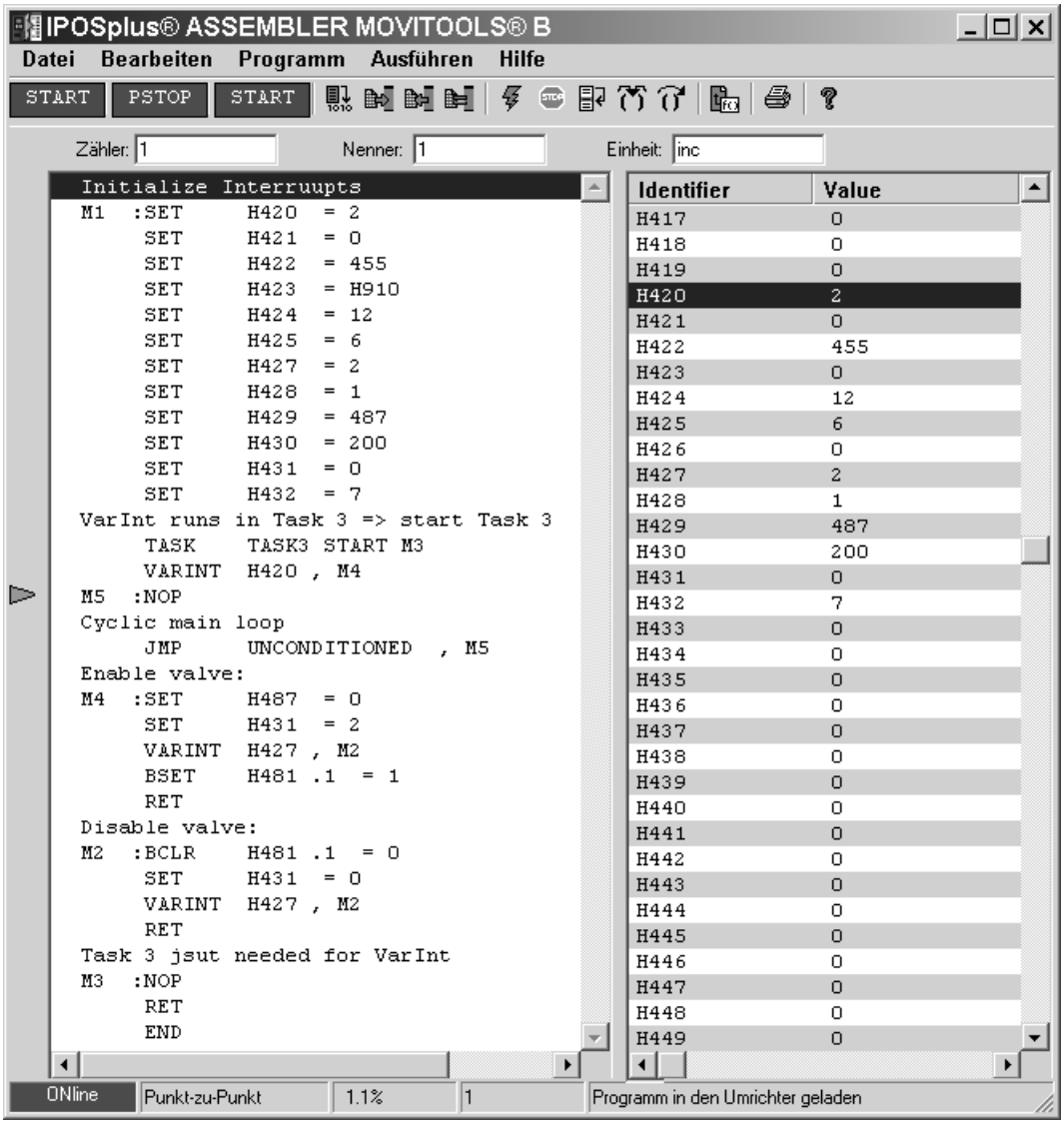


Figura 1: Assembler

5 Detecção da Posição e Posicionamento

5.1 Introdução

Posicionamento IPOS$^{plus®}$

Pode-se utilizar o controle de posicionamento IPOS$^{plus®}$ somente quando um encoder do motor estiver conectado em X15 e um modo de operação ".. & IPOS" estiver ajustado em P700 Operating mode 1. P941 Source actual position determina qual medida de posição é utilizada para posicionamento. Os comandos de percurso do controle IPOS$^{plus®}$ (comandos GO) referem-se a informação de posição do encoder inserido em P941 Source actual position.

5.2 Avaliação do encoder

O MOVIDRIVE® oferece vários opcionais para posicionamento:

- Encoder externo
- Encoder do motor (encoder incremental/resolver)
- Encoder Hiperface® (encoder absoluto)
- Encoder absoluto SSI

Os valores são fornecidos para processamento nas variáveis do sistema.

As conexões para encoder do motor (X15) e encoder externo (X14) estão nas eletrônicas de controle Mx**V**..., Mx**S**... e MC**H**... . As eletrônicas de controle Mx**F** não têm estas conexões. A conexão para o encoder absoluto SSI está na placa opcional DIP11 (X62).

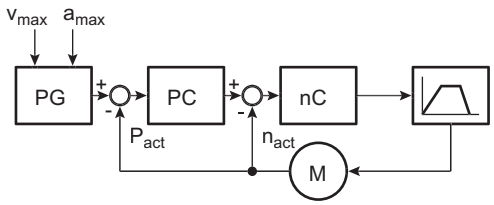
Todos os encoders conectados são sempre avaliados independente do modo de operação (P700). Os modos de operação com posicionamento (VFC-n-CTRL & IPOS, CFC & IPOS, SERVO & IPOS) sempre necessitam um encoder do motor em X15.

Tipo do encoder		**Encoder absoluto na DIP11 P941: Encoder absoluto (DIP)**	**Encoder Hiperface® / simulação do encoder incremental / encoder incremental P941: Encoder externo (X14)**	**Encoder incremental/ resolver / Encoder Hiperface® P941: Encoder do motor (X15)**
Conexão		X62 / DIP11	X14 / unidade básica	X15 / unidade básica
Valor atual na variável		H509 / ACTPOS. ABS / ActPos_Abs	H510 / ACTPOS. EXT / ActPos_Ext	H511 / ACTPOS. MOT / ActPos_Mot
Resolução		Posição absoluta **após** conversão com: Encoder scaling (P955), Zero offset (P954), Position offset (P953), Counting direction (P951).	**Número de pulsos por volta do encoder atual** (multiplicado por 4) **após** conversão com: Encoder scaling ext. encoder (P944)	**Sempre** 4096 increm./volta do motor, independente da resolução atual do encoder
Touch probe	**Pulso em DI02**	H503 / TP. POS1ABS / TpPos1_Abs	H506 / TP. POS1EXT / TpPos1_Ext	H507 / TP. POS1MOT / TpPos1_Mot
	Pulso em DI03	H502 / TP. POS2ABS / TpPos2_Abs	H504 / TP. POS2EXT / TpPos2_Ext	H505 / TP. POS2MOT / TpPos2_Mot
	Tempo máx. de atraso	1 ms	< 100 µs	< 100 µs

Os valores de posição são sempre disponíveis para controle IPOS$^{plus®}$ nas variáveis H509 a H511. Mesmo se o posicionamento foi realizado sem IPOS$^{plus®}$, os encoders de pulso conectados ao X14 e X15 podem ser gravados e além disso, processados no programa IPOS$^{plus®}$. Um encoder do motor deve ser utilizado para posicionamento com os comandos IPOS$^{plus®}$ (GO...). O encoder do motor fornece o MOVIDRIVE® com um sinal de rotação de alta qualidade.

5.3 Combinações do encoder

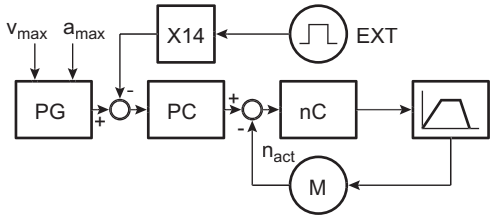
Controle de posição direto com encoder do motor



- Deve ser instalado no motor um encoder incremental / resolver / Encoder Hiperface® (X15).
- No IPOS$^{plus®}$, os comandos de posicionamento, por exemplo, "GOA ..." são realizados com relação a fonte atual de posição (neste caso, encoder do motor X15).

v_{max} = Rotação máxima
a_{max} = Aceleração máxima
PG = Gerador do perfil
P_{act} = Posição atual do encoder do motor
PC = Controlador de posição
n_{act} = Rotação atual
nC = Controlador de rotação

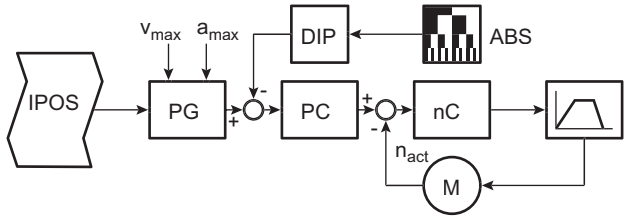
Controle de rotação direto com encoder externo e encoder do motor



- Sempre é necessário no motor um encoder incremental / resolver / Encoder Hiperface® (X15) para realimentação da rotação.
- O escorregamento ou a folga mecânica (folga angular do redutor) entre o encoder do motor e o encoder externo é compensado automaticamente.
- No IPOS$^{plus®}$, os comandos de posicionamento, por exemplo, "GOA ..." são realizados com relação a P941 Source actual position (neste caso, encoder do motor X14).
- A reação dinâmica que pode ser obtida depende das propriedades e da instalação mecânica do encoder externo assim como a resolução de posição.
- Ver a seção "IPOS$^{plus®}$ com Opcionais" / "Encoder externo".

v_{max} = Rotação máxima
a_{max} = Aceleração máxima
PG = Gerador do perfil
PC = Controlador de posição
n_{act} = Rotação atual
nC = Controlador de rotação
EXT = Encoder externo

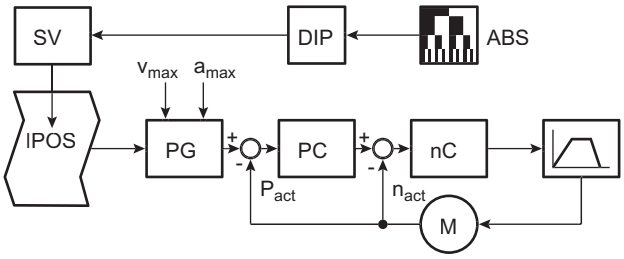
Controle de posição direto com encoder absoluto e encoder do motor



- Controle de posição direto no IPOS$^{plus®}$ por meio do encoder absoluto SSI conectado através da DIP11.
- Sempre é necessário no motor um encoder incremental / resolver / Encoder Hiperface® (X15) para realimentação da rotação.
- O escorregamento ou a folga mecânica (folga angular do redutor) entre o encoder incremental / resolver / Hiperface® do motor e o encoder absoluto é compensado automaticamente.
- No IPOS$^{plus®}$, os comandos de posicionamento, por exemplo, "GOA ..." são realizados com relação a fonte atual de posição (neste caso, encoder absoluto DIP).
- A reação dinâmica que pode ser obtida depende das propriedades e da instalação do encoder absoluto, assim como a resolução de posição.
- Consultar o manual "Positioning with Absolute Encoder DIP11A".

v_{max} = Rotação máxima
a_{max} = Aceleração máxima
PG = Gerador do perfil
PC = Controlador de posição
n_{act} = Rotação atual
nC = Controlador de rotação
ABS = Encoder absoluto
IPOS = Programa IPOS$^{plus®}$

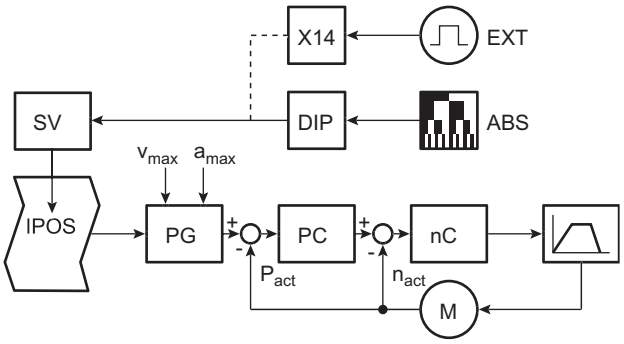
Controle de posição com encoder incremental no motor,
Processamento da posição do encoder absoluto no programa IPOS$^{plus®}$



- O controle de posição é realizado no IPOS$^{plus®}$ utilizando o encoder do motor conectado ao X15.
- Sempre é necessário no motor um encoder incremental / resolver para realimentação da rotação.
- A alta reação dinâmica do conversor pode ser utilizada diretamente para posicionamento.
- A informação de posição do encoder absoluto é mapeada automaticamente em uma variável IPOS$^{plus®}$ e pode ser processada utilizando o controle do programa.
- Utilizando a DIP11 nesta maneira significa que o referenciamento é desnecessário.
- Consultar o manual "Positioning with Absolute Encoder DIP11."

v_{max} = Rotação máxima
a_{max} = Aceleração máxima
PG = Gerador do perfil
P_{act} = Posição atual do encoder do motor
PC = Controlador de posição
n_{act} = Rotação atual
nC = Controlador de rotação
ABS = Encoder absoluto
SV = Variável do sistema
IPOS = Programa IPOS$^{plus®}$

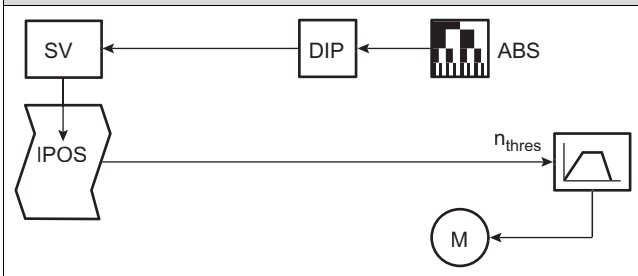
Controle de posição com encoder do motor,
Processamento do segundo encoder no programa IPOS$^{plus®}$ como encoder mestre



- O controle de posição é realizado no IPOS$^{plus®}$ utilizando o encoder do motor conectado ao X15.
- Sempre é necessário no motor um encoder para realimentação da rotação.
- A alta reação dinâmica do conversor pode ser utilizada diretamente para posicionamento.
- A informação de posição do segundo encoder é mapeada automaticamente em uma variável IPOS$^{plus®}$ e pode ser processada utilizando o controle do programa.
- Esta estrutura é utilizada quando o conversor funciona com um segundo encoder (por exemplo, ângulo síncrono ou came eletrônico)

v_{max} = Rotação máxima
a_{max} = Aceleração máxima
PG = Gerador do perfil
P_{act} = Posição atual do encoder do motor
PC = Controlador de posição
n_{act} = Rotação atual
nC = Controlador de rotação
ABS = Encoder absoluto
EXT = Encoder externo
SV = Variável do sistema
IPOS = Programa IPOS$^{plus®}$

Processamento da posição do encoder absoluto no programa IPOS$^{plus®}$



- A informação de posição do encoder absoluto é mapeada automaticamente em uma variável IPOS$^{plus®}$ e pode ser processada utilizando o controle do programa.
- A DIP11 ou um encoder HIPERFACE® em X14 pode ser utilizado especialmente para substituir aplicações em que o posicionamento normalmente ocorre utilizando velocidade rápida/velocidade reduzida por meio de várias chaves de proximidade.
- Não é necessário nenhum encoder incremental/resolver no motor para realimentação da rotação; pode ser utilizado um motor assíncrono padrão. Não é necessário o modo de operação "... & IPOS".
- Consultar o manual "Positioning with Absolute Encoder DIP11". O módulo aplicativo "Absolute positioning" é disponível no SHELL para realizar este tipo de tarefa de posicionamento.

ABS = Encoder absoluto
SV = Variável do sistema
IPOS = Programa IPOS$^{plus®}$
n_{thres} = Rotação de ajuste

5.4 Encoder externo (X14)

Os encoders a seguir podem ser conectados aos conversores de freqüência MOVIDRIVE®A e B em X14. Para descobrir o tipo de encoder que o seu equipamento suporta, consultar o manual de sistema ou as instruções de operação.

- Encoder Hiperface® tipo AS1H, ES1H ou AV1H (somente MOVIDRIVE® B ou MCH)
- Encoder sen/cos tipo ES1S, ES2S ou EV1S (somente MOVIDRIVE® B ou MCH)
- Sensor TTL 5 V com tensão de alimentação 24 V_{CC} tipo ES1R, ES2R ou EV1R
- Sensor TTL 5 V com tensão de alimentação 5 V_{CC} tipo ES1T, ES2T ou EV1T via opcional DWI11

Posicionamento com encoder externo (X14)

Faz sentido utilizar um encoder externo para posicionamento, para compensar qualquer conexão sujeita a escorregamento ou distância da folga entre o conversor (por exemplo, devido as rodas de deslocamento ou folga angular dos dentes) ou qualquer folga mecânica no redutor.

Tabela 1: Evolução dos pulsos do encoder externo

Pulsos de entrada (exemplo)	2048	2048	1024	1024
Avaliação quádrupla (fixo)	8192	8192	4096	4096
Escala do encoder externo P944 (pode ser ajustado)	x 1	x 8	x 1	x 2
Mudando a leitura do contador H510 ACTPOS. EXT por volta do encoder	**8192**	**65536**	**4096**	**8192**

Uma vez a tensão de controle ter sido ligada, se for utilizado um AS1H ou AV1H o valor absoluto deste encoder é utilizado como o valor de posição atual do encoder externo H510. Para todos os outros tipos de encoder, H510 = 0 incrementos. O encoder externo pode ser referenciado como encoder do motor (ver o capítulo Referenciamento).

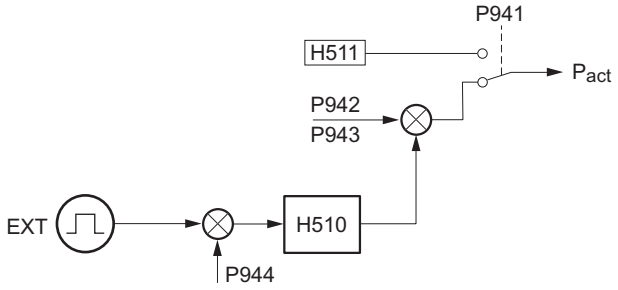
Compensação de escorregamento com encoder externo

Um transportador sobre rodas funciona em trilhos. O carro é movido pela movimentação das rodas com um motoredutor. A conexão entre as rodas e os trilhos é não-positiva. Isto causa escorregamento entre o movimento rotacional da roda e o movimento translacional resultante do carro.

isto quer dizer que, para posicionamento utilizando controle do motor é necessário que a posição do carro seja identificada.

Para isto, na colocação em operação a relação de incrementos do encoder do motor deve ser ajustada para os incrementos do encoder síncrono. Uma descrição destes parâmetros com exemplos, e o procedimento para ajustá-los podem ser encontrados nas descrições dos parâmetros P944, P943 e P942 na seção 'Parâmetros IPOS.'

Aplica-se o seguinte esquema de ligação:



P_x = Valor de posição não-linearizada do encoder
P_{act} = Valor de posição atual para gerador de rampa e controlador de posição
P941 = Fonte de posição atual
P942 = Fator numerador do encoder
P943 = Fator denominador do encoder
P944 = Escala do encoder externo
H510 = Encoder externo com valor atual na variável
H511 = Encoder do motor com valor atual na variável

Ajustar os seguintes parâmetros para o encoder externo:

Tabela 2: Ajustes de parâmetros para o transportador

Nr.	Denominação	Função	Ajuste	Faixa
P944	Escala do encoder externo	Multiplica os sinais do encoder com o valor ajustado	Valor mais alto que é menor do que a relação entre o número de pulsos por volta do encoder do motor e o encoder externo. Exemplo: Encoder do motor: 4096 Inc./ encoder externo 800 inc. = 5.12. Valor: 4.	Fixo: 1, 2, 4, 8, 16, 32, 64
P943	Fator denominador do encoder	Denominador para determinar a relação entre o encoder do motor e o encoder externo.	Número de incrementos (em H511, para ler o ACTPOS. MOT) para uma certa distância s.	Máx. 32767
P942	Fator numerador do encoder	Numerador para determinar a relação entre o encoder do motor e o encoder externo.	Número de incrementos (em H510, para ler o ACTPOS. EXT) para uma certa distância s, como para P943.	Máx. 32767
P941	Fonte de posição atual	Valor de posição atual para controle de posicionamento IPOS$^{plus®}$	Encoder externo X14	(Seleção)
P945	Tipo do encoder síncrono (X14)	Seleciona o tipo do encoder	Depende do encoder que é conectado.	TTL SIN/COS HIPERFACE
P946	Sentido de contagem do encoder síncrono (X14)	Inverte o sentido de rotação do encoder	Ajustar de modo que o sentido de contagem do encoder do motor = sentido de contagem do encoder externo.	NORMAL INVERTED

O cálculo de P210 (Controlador de retenção ganho P) é otimizado para P941 = encoder do motor na colocação em operação. Se utilizar um encoder externo ou encoder absoluto, o parâmetro deve ser ajustado para um valor inferior.

O seguinte aplica-se para detecção da posição com um encoder externo em X14:

- Variável H510 mostra a posição atual do controle de posição ACTPOS. EXT
- Variável H506 mostra a posição touch probe 1 TP. POS1EXT
- Variável H504 mostra a posição touch probe 2 TP. POS2EXT

As variáveis são sempre avaliadas com os parâmetros P944, P943 e P942.

5.5 Encoder absoluto SSI (DIP)

Colocação em operação para DIP com encoder absoluto

O sistema deve ser iniciado em conjunto com o conversor de freqüência MOVIDRIVE® como descrito no manual de sistema do MOVIDRIVE®. Isto deve ser possível para movimentar o conversor utilizando setpoint e fonte do sinal de controle adequados.

Além disso, assegurar que as seguintes instalações estejam corretas e utilizadas como especificado:

- Instalação da DIP11A / DIP11B
- Cabeamento
- Programação dos bornes
- Interruptores de segurança

Não há necessidade de ativar os ajustes de fábrica. Se acessar um ajuste de fábrica, os parâmetros do MOVIDRIVE® serão reativados para os valores padrão. Isto também afeta a programação dos bornes, que devem ser alterados, se necessário, para os ajustes desejados.

O MOVITOOLS® orienta através do procedimento de colocação em operação do encoder absoluto opcional. Várias caixas de diálogo alertam a fazer as entradas necessárias e tomar as ações desejadas. Para isto, iniciar o MOVITOOLS® Manager e estabelecer uma conexão com o conversor selecionando a interface e o endereço da unidade. Então rodar o programa SHELL e iniciar a colocação em operação através do menu [Startup] / [Startup for / DIP]. As caixas de diálogo fornecem toda a ajuda necessária para o procedimento restante.

Uma vez a colocação em operação com MOVITOOLS® ter sido completada, especificar somente o parâmetro "Source actual position".

Por outro lado, pode-se realizar a colocação em operação do opcional DIP11 para o encoder absoluto, passo-a-passo como descrito a seguir. O mesmo procedimento pode ser utilizado para o controle opcional DBG11. Se ocorrer a mensagem de irregularidade F92 "DIP registration range" durante a colocação em operação, reconheça-a com um reset e continue com o procedimento de colocação em operação. Esta mensagem não deve aparecer, uma vez a colocação em operação ter sido realizada com sucesso.

***1.** Selecionar o tipo do encoder P950*

No P950 Encoder type, selecionar o encoder que está utilizando. Os sistemas de encoder realizados são indicados na descrição de P950.

O tipo conectado pode ser selecionado a partir da lista dos possíveis encoders. Para outros encoders deve ser verificada a sua adequação e devem ser autorizados pela SEW-EURODRIVE.

***2.** Ajustar o sentido de rotação para o motor P35_*

Movimentar o conversor no sentido positivo (definido conforme a aplicação) em baixa rotação. Se a contagem da posição atual P003 ou H511 aumentar, não mudar o parâmetro P350 "Change direction of rotation" (utilizar MOVITOOLS® ou DBG11B para indicar a posição atual). Mudar P350 se a contagem da posição atual diminuir. Isto adapta o sentido de contagem do encoder do motor para a aplicação.

***3.** Ajustar o sentido de contagem P951 para o encoder absoluto SSI*

Movimentar o conversor no sentido positivo (definido conforme a aplicação) em baixa rotação. Se a contagem da posição do encoder absoluto (H509 ACTPOS. ABS) aumentar, não mudar o parâmetro P951 "Counting direction". Se a contagem da posição do encoder absoluto diminuir, P951 deve ser invertido.

***4.** Ajustar a escala do encoder P955*

Este parâmetro é irrelevante a não ser que esteja presente um encoder do motor (controle de rotação). A informação da posição do encoder absoluto é multiplicada pelo valor ajustado. O parâmetro é ajustado de modo que a relação da informação de deslocamento entre o encoder do motor e o encoder absoluto seja mais próxima possível de "1".

Com objetivos de medição, ajustar o parâmetro para 1 no início. Anotar os valores das variáveis H509 (ACTPOS. ABS) e H511 (ACTPOS. MOT). Movimentar o conversor por pelo menos 1 volta do motor. Determinar a diferença entre o valor anotado e o valor atual das variáveis e calcular o quociente.

ACTPOS. ABS	H509 antigo	H509 novo	Diferença da H509
Valor anotado			
ACTPOS. MOT	H511 antigo	H511 novo	Diferença da H511
Valor anotado			

O quociente Q resulta da diferença da H511 dividido pela diferença da H509.

Q = (H511 antigo – H511 novo) / (H509 antigo – H509 novo)

Ajustar o parâmetro ENCODER SCALING (P955) para o valor mais próximo do quociente Q calculado, de preferência para o valor mais baixo.

Se o quociente for maior do que 80, o posicionamento utilizando o encoder absoluto somente pode ser realizado com propriedades dinâmicas reduzidas.

Durante o planejamento de projeto, assegure que a relação do encoder não excede 1:10.

***5.** Ajustar o offset de posição P953*

O offset de posição (P953) deve ser ajustado somente para encoders incrementais. Para outros encoders, deve ser ajustado para 0.

Continuar conforme descrito para P953 na seção 'Parâmetros IPOS.'

***6.** Ajustar offset Zero P954*

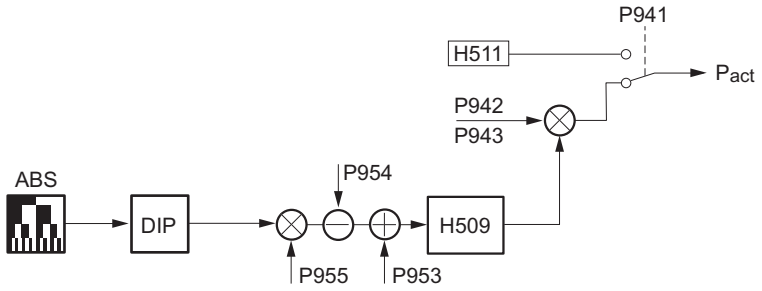
O offset Zero é utilizado para fixar o valor desejado para uma posição específica. Inserir o parâmetro conforme descrito para P954 na seção 'Parâmetros IPOS.'

***7.** Ajustar fatores do encoder P942 e P943*

Os parâmetros são utilizados para ajuste interno do controle de rotação e para funções de monitoração na DIP11.

Praticamente, este adapta a quantidade física, uma relação mecânica entre o encoder do motor e o encoder externo e a constante taxa de alimentação mecânica (por exemplo, para encoder incremental externo).

O diagrama a seguir mostra a conexão entre os parâmetros e as variáveis.



P_x = Valor de posição não-linearizada do encoder
P_{abs}= Valor de posição atual para gerador de rampa e controlador de posição
P941 = Fonte de posição atual
P942 = Fator numerador do encoder
P943 = Fator denominador do encoder
P953 = Offset de posição
P954 = Offset Zero
P955 = Escala do encoder
H509 = Encoder externo com valor atual na variável
H511 = Encoder do motor com valor atual na variável

Para descobrir como determinar os fatores do encoder, consultar as descrições dos parâmetros para P942 / P943.

***8.** Ajustar a fonte de posição atual P941*

Este parâmetro determina qual posição do encoder é utilizada para controle de posição desde que o modo de operação "... & IPOS" tenha sido ajustado no parâmetro P700, "Operating mode".

O programa IPOS$^{plus®}$ contém comandos de posicionamento para controlar o motor conectado ao MOVIDRIVE®. Ajustar a "Fonte de posição atual" para "Encoder absoluto DIP" se o motor for posicionado utilizando o encoder absoluto.

O ganho do circuito para controle de posição do IPOS$^{plus®}$, parâmetro P910 "Gain X controller" foi pré-definido durante a colocação em operação do circuito de controle de rotação. Este pré-ajuste significa que o controle de posicionamento é realizado com o encoder do motor. A diferença no número de pulsos por volta do encoder ou as características de tempo do encoder absoluto (por ex. instrumento de medição de distância à laser) pode necessitar um ajuste de valor inferior.

Ajustar no máximo a metade do valor pré-definido calculado. Se P955 é ≥ 32, inserir somente um quarto do valor pré-definido calculado. Iniciar um programa IPOS$^{plus®}$ com uma operação de posicionamento entre dois pontos válidos em rotação moderada. Reduzir ou aumentar o parâmetro P910 "Gain X controller" passo-a-passo até o movimento e as características de posicionamento terem sido ajustados. Se P955 é ajustado para um valor alto, pode ser necessário que os valores em P910 sejam < 1.

O valor de posição fornecido pelo encoder absoluto está disponível na variável H509 (ACTPOS. ABS). O valor de posição pode ser processado com o controle interno IPOS$^{plus®}$ mesmo sem posicionamento direto.

5.6 Referenciamento

Nas aplicações utilizando comandos de posicionamento absoluto, deve-se definir o ponto de referência (zero da máquina). Dependendo do tipo de encoder, este ajuste deve ser feito na colocação em operação inicial (encoder absoluto) ou cada vez que a máquina for novamente ligada (todos os outros encoders).

O MOVIDRIVE® realiza 9 tipos de referenciamento que podem ser ajustados através do P903 Tipo de referenciamento, P904 Referenciamento para pulso zero e os argumentos do comando IPOS$^{plus®}$ `_Go0 ( ... )` ou `G00 ...`. Eles são constituídos de uma combinação adequada das seguintes propriedades:

- O encoder é ajustado sem referenciamento
- Sentido de busca (= sentido do movimento no início do referenciamento)
- Referenciamento para as chaves fim-de-curso do hardware
- Referenciamento para pulso zero (possível somente com encoders externos quando o número de pulsos < 5000 inc./volta)
- Referenciamento para o came de referência

Se as chaves fim-de-curso do hardware e / ou o came de referência forem utilizados para referenciamento, eles devem ser ajustados como entradas digitais.

O parâmetro P941 Source actual position é utilizado para definir qual encoder é utilizado para referenciamento.

Uma vez o conversor estar liberado, o referenciamento é iniciado utilizando um dos seguintes métodos:

- Através de um flanco positivo na entrada digital P600 ... P606 ou P610 ... P617 que é ajustado para a função REF. TRAVEL START
- Através do comando IPOS$^{plus®}$ `_Go0 ( ... )` ou `G00 ...`

O display muda para "c" - REFERENCE MODE.

A rampa de parada P136 sempre é utilizada para aceleração durante o referenciamento. Nos tipos de referenciamento com came de referência ou chave fim-de-curso, primeiro o conversor é acelerado para a rotação ajustada em P901 Reference speed 1 e então o sistema busca a condição para terminar o referenciamento com P902 Reference speed 2. Para o tipo de referência 0, é utilizado imediatamente P902 Reference speed 2.

Se é alcançada uma chave fim-de-curso do hardware durante referenciamento com tipo 1 ou tipo 2 e o ponto de referência ainda não tiver sido encontrado, o conversor gira e continua o referenciamento no outro sentido.

Uma vez o conversor ter encontrado o ponto de referência, são realizadas as seguintes funções:

- O conversor pára e comuta internamente da rotação para o controle de posição.
- O Bit 20 "IPOS Referenced" em H473 StatusWord é ajustado e uma entrada digital, com o parâmetro "IPOS REFERENCE", é ajustada.
- O offset de referência P900 é transferido para o valor de posição atual. Se a função módulo é desativada (P960 = OFF), esta é a variável do encoder H509 - H511 selecionada em P941. Se a função módulo é ativada, esta é a variável H455 ModActPos.

Deste ponto, a fórmula a seguir aplica-se para o zero da máquina:

Zero da máquina = ponto de referência + offset de referência

O estado "Referenced" é reativado quando o conversor é desligado ou se ocorrer uma mensagem de irregularidade relativa ao sistema de medição de posição (exceção: para encoders Hiperface®, ver a informação abaixo).

Para usuários avançados, o MOVIDRIVE® também oferece a opção de ajuste de encoders absolutos pelo cálculo e descrevendo um novo offset de encoder no ponto de referência sem o conversor ser liberado e enquanto ele está, por exemplo, em estado de parada segura. Além disso, isto também é possível para avaliar os sistemas de encoder codificados à distância utilizando programação complexa IPOS$^{plus®}$. Se desejar suporte na realização destas operações, consultar a SEW-Service.

Para encoders absolutos Hiperface® e encoders absolutos SSI, o estado "Referenced" sempre é ajustado e é reativado somente durante o referenciamento.

O conversor reage de maneira diferente quando o regulador bloqueado é ajustado durante o referenciamento, dependendo de como o referenciamento foi iniciado.

- Se o referenciamento foi iniciado através de um flanco positivo na entrada digital REF.TRAVEL START ou através de um comando sem espera IPOS `_Go0( ._NW_.. )`; ou `GO0...,NW,...`, o referenciamento não é continuado quando o regulador bloqueado é retirado. Para iniciar o percurso novamente, deve ser ajustado um flanco positivo na entrada REF. TRAVEL START.
- Se o referenciamento foi iniciado através de um comando com espera IPOS `_Go0 ( ._W_.. )`; ou `GO0...,W,...`, é gerada a mensagem de irregularidade F39 Reference travel.

Quando decidir a referência para o came de referência ou pulso zero, observar os seguintes pontos:

- O pulso zero muda quando o motor é substituído.
- O came de referência pode tornar-se impreciso com o passar do tempo, desgaste ou histereses de comutação.
- Se o ponto de referência é determinado utilizando pulso zero e came de referência, e o pulso zero é localizado exatamente no final do came de referência, a transição de comutação do came de referência pode ser identificada antes ou depois do pulso zero (histereses de comutação). O resultado pode ser uma posição de referência que varia por uma volta do motor eventualmente. A situação pode ser remediada deslocando o came de referência (cerca de metade de uma volta do motor).
- Os sistemas de um sentido somente podem ser referenciados utilizando um came de referência. Adicionalmente, observar que não há uma distância definida entre o came de referência e o pulso zero do encoder para relações não-inteiras. Isto significa que neste caso somente o final do came de referência pode ser selecionado como ponto de referência.
- O comprimento do came de referência e as rotações de referência devem ser selecionados para o conversor poder desacelerar com segurança para a rotação de referência mais lenta (rotação de referência 2) no came de referência. O final do came de referência ou o pulso zero mais próximo do sistema de encoder pode ser utilizado como ponto de referência.
- O pulso zero somente pode ser utilizado para referenciamento quando o encoder tiver pulso zero, o sinal zero é conectado ao conversor e o número de pulsos por volta do encoder < 5000 inc./volta.

Para referenciamento de um conversor com encoder absoluto (Hiperface® ou DIP), o offset de posição será recalculado e regravado pelo referenciamento P905 Hiperface® offset X14 / P947 Hiperface® offset X15 ou DIP offset P953 Position offset, dependendo da fonte ajustada para a posição atual.

A seguir, os diferentes tipos de referenciamento com diferentes pontos de partida no conversor, utilizando diagramas de percurso.

Explicação dos diagramas do tipo de referenciamento

- nRef1 =Rotação de referência 1
- nRef2 =Rotação de referência 2
- Ponto de partida do conversor
  - [1] Entre o came de referência e a chave fim-de-curso horária do hardware
  - [2] No came de referência
  - [3] Entre o came de referência e a chave fim-de-curso antihorária do hardware
- LHWLS = Chave fim-de-curso antihorária do hardware
- RHWLS = Chave fim-de-curso horária do hardware
- CAM = Came de referência
- RefCAM = Came da posição de referência: O movimento para esta posição ocorre quando o argumento do comando de referenciamento GO0 é ajustado para **CAM**.
- RefZP = Pulso zero da posição de referência: O movimento para esta posição ocorre quando o argumento do comando de referenciamento GO0 é ajustado para **ZP**.
- RefOffCAM = Offset de referência para referenciamento com came da posição de referência CAM
- RefOffZP = Offset de referência para referenciamento com pulso zero ZP
- MZP = Zero da máquina

Tipo 0: Referenciamento para pulso zero

A posição de referência é o primeiro pulso zero antihorário da posição de partida do referenciamento.

Não é necessário um came de referência. Somente P902 Reference speed 2 é utilizado para referenciamento.

Se o referenciamento é iniciado através do flanco positivo na entrada "REF.TRAVEL START", P904 Referenciamento para pulso zero deve ser ajustado para YES.

Se o referenciamento é iniciado com o comando IPOS$^{plus®}$ `Go0`, o argumento `"ZP"` deve ser ajustado; P904 não tem importância.

No primeiro caso, se P904 = NO ou se o argumento `"CAM"` é utilizado no segundo caso, o conversor age como para o tipo 5 e ajusta a posição atual para a posição de referência.

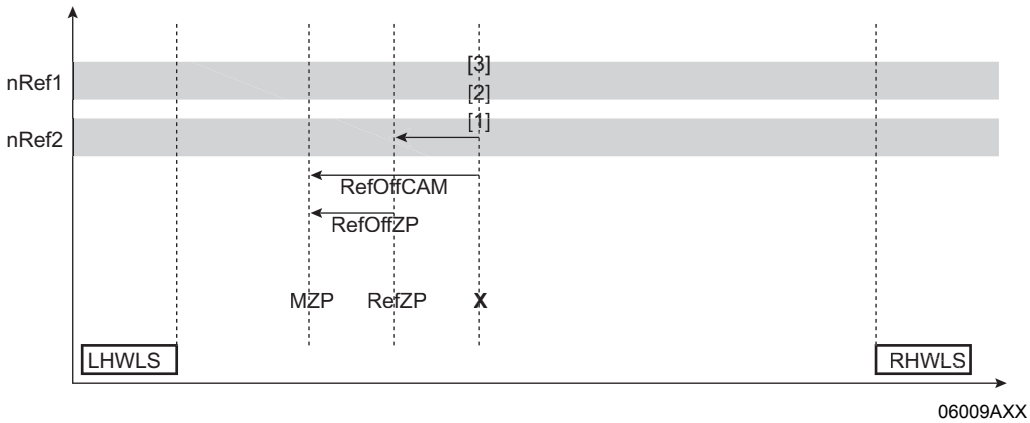


Figura 2: Tipo de referenciamento 0 / X = posição atual do conversor

Tipo 1: Extremidade esquerda do came de referência

A posição de referência é a extremidade esquerda do came de referência ou o primeiro pulso zero da esquerda após o final do came de referência.

Uma das entradas digitais P600 ... 606 deve ser ajustada para "REFERENCE CAM."

O referenciamento inicia no sentido antihorário; P901 Reference speed 1 é utilizado até o primeiro flanco positivo do came de referência, então é utilizado P902 Reference speed 2.

Se o referenciamento é iniciado através do flanco positivo na entrada "REF.TRAVEL START", o conversor é referenciado na queda de flanco do came de referência ou no pulso zero após a queda de flanco do came de referência, dependendo do ajuste no P904 Referencing to zero pulse.

Se o referenciamento é iniciado com o comando IPOS$^{plus®}$ `Go0`, o conversor é referenciado na queda de flanco do came de referência ou no pulso zero após a queda de flanco do came de referência, dependendo se o argumento `"ZP"` ou `"CAM"` é ajustado.

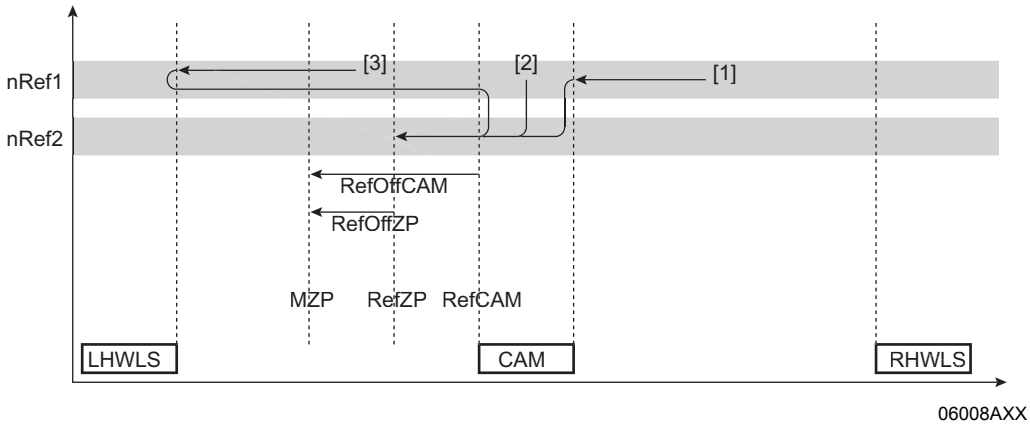


Figura 3: Referenciamento tipo 1

Tipo 2: Extremidade direita do came de referência

A posição de referência é a extremidade direita do came de referência ou o primeiro pulso zero da direita após o final do came de referência.

Uma das entradas digitais P600 ... 606 deve ser ajustada para "REFERENCE CAM."

O referenciamento inicia no sentido horário; P901 Reference speed 1 é utilizado até o primeiro flanco positivo do came de referência, então é utilizado P902 Reference speed 2.

Se o referenciamento é iniciado através do flanco positivo na entrada "REF.TRAVEL START", o conversor é referenciado na queda de flanco do came de referência ou no pulso zero após a queda de flanco do came de referência, dependendo do ajuste no P904 Referencing to zero pulse

Se o referenciamento é iniciado com o comando IPOS$^{plus®}$ `Go0`, o conversor é referenciado na queda de flanco do came de referência ou no pulso zero após a queda de flanco do came de referência, dependendo se o argumento `"ZP"` ou `"CAM"` é ajustado.

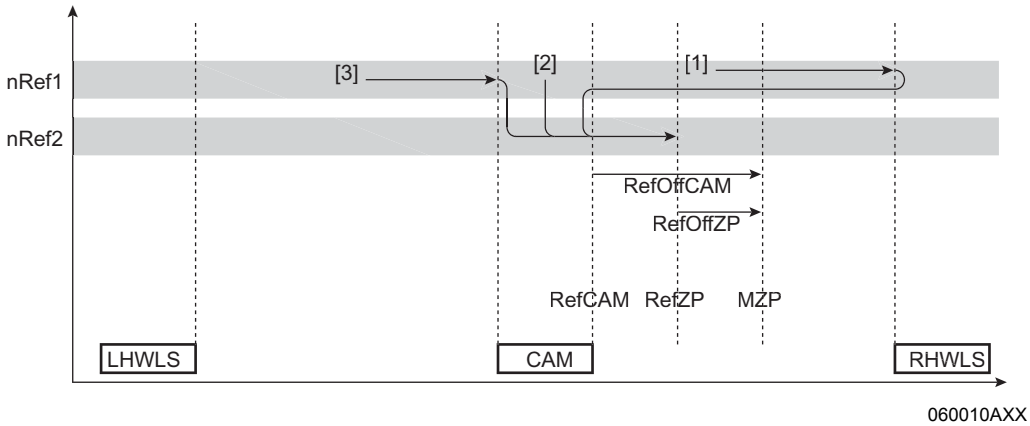


Figura 4: Referenciamento tipo 2

Tipo 3: Chave fim-de-curso horária

O ponto de referência é o primeiro pulso zero à esquerda da chave fim-de-curso horária.

O ajuste "Left end of the CW limit switch" não é importante porque após o referenciamento, o conversor deve estar localizado na histerese da chave fim-de-curso e a irregularidade "29 Limit switch reached" pode ocorrer esporadicamente, uma vez o referenciamento estar completo. Não é necessário um came de referência.

O referenciamento inicia no sentido horário. P901 Reference speed 1 é utilizado até a queda do flanco da chave fim-de-curso horária, então é utilizado P902 Reference speed 2.

Se o referenciamento é iniciado através do flanco positivo na entrada "REF.TRAVEL START", P904 Referenciamento para pulso zero deve ser ajustado para YES.

Se o referenciamento é iniciado com o comando IPOS$^{plus®}$ `Go0`, deve ser ajustado o argumento `"ZP"`.

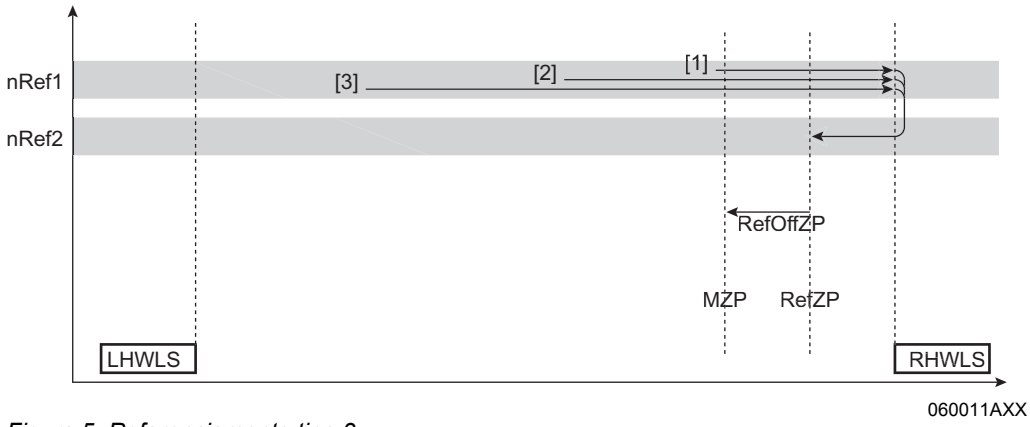


Figura 5: Referenciamento tipo 3

Tipo 4: Chave fim-de-curso antihorária

O ponto de referência é o primeiro pulso zero à direita da chave fim-de-curso antihorária.

O ajuste "Right end of the CCW limit switch" não é importante porque após o referenciamento, o conversor deve estar localizado na histerese da chave fim-de-curso e a irregularidade "29 Limit switch reached" pode ocorrer esporadicamente, uma vez o referenciamento estar completo. Não é necessário um came de referência.

O referenciamento inicia no sentido antihorário; P901 Reference speed é utilizado até a queda do flanco da chave fim-de-curso antihorária, então é utilizado P902 Reference speed 2.

Se o referenciamento é iniciado através do flanco positivo na entrada "REF.TRAVEL START", P904 Referenciamento para pulso zero deve ser ajustado para YES.

Se o referenciamento é iniciado com o comando IPOS$^{plus®}$ `Go0`, deve ser ajustado o argumento `"ZP"`.

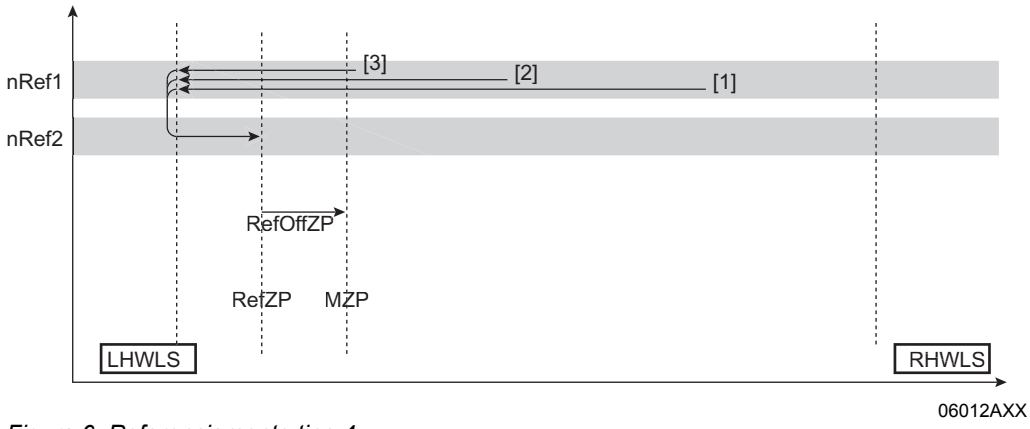


Figura 6: Referenciamento tipo 4

Tipo 5: Sem referenciamento

A posição de referência é a posição atual. Os argumentos no comando IPOS$^{plus®}$ `Go0` `"ZP"` ou `"CAM"` e P904 não têm função.

Faz sentido utilizar este tipo de referenciamento com encoders absolutos e para sistemas que precisam ser referenciados parados. Por exemplo, a posição de um eixo pode ser ajustada para zero quando o conversor está parado. Deste modo, o operador da máquina pode dizer onde o conversor está localizado dentro de cada movimento.

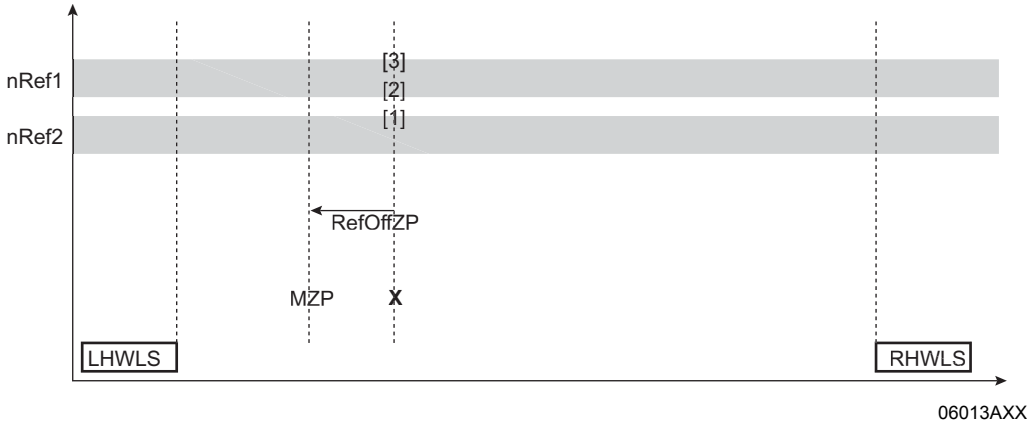


Figura 7: Referenciamento tipo 5

Tipo 6: Came de referência alinhado com a chave fim-de-curso horária

A posição de referência é a extremidade esquerda do came de referência ou o primeiro pulso zero da esquerda após o final do came de referência.

Uma das entradas digitais P600 ... 606 deve ser ajustada para "REFERENCE CAM." O referenciamento inicia no sentido horário; P901 Reference speed 1 é utilizado até o primeiro flanco positivo do came de referência, então é utilizado P902 Reference speed 2. Ao contrário do tipo 1, o conversor inicia no sentido horário e gira no came de referência.

Se o referenciamento é iniciado através do flanco positivo na entrada "REF.TRAVEL START", o conversor é referenciado na queda de flanco do came de referência ou no pulso zero após a queda de flanco do came de referência, dependendo do ajuste no P904 Referencing to zero pulse.

Se o referenciamento é iniciado com o comando IPOS$^{plus®}$ `Go0`, o conversor é referenciado na queda de flanco do came de referência ou no pulso zero após a queda de flanco do came de referência, dependendo se o argumento `"ZP"` ou `"CAM"` é ajustado.

O came de referência deve iniciar antes ou de acordo com a chave fim-de-curso horária do hardware e deve projetar para a chave fim-de-curso. Isto garante que nenhum contato seja feito com a chave fim-de-curso do hardware durante o referenciamento.

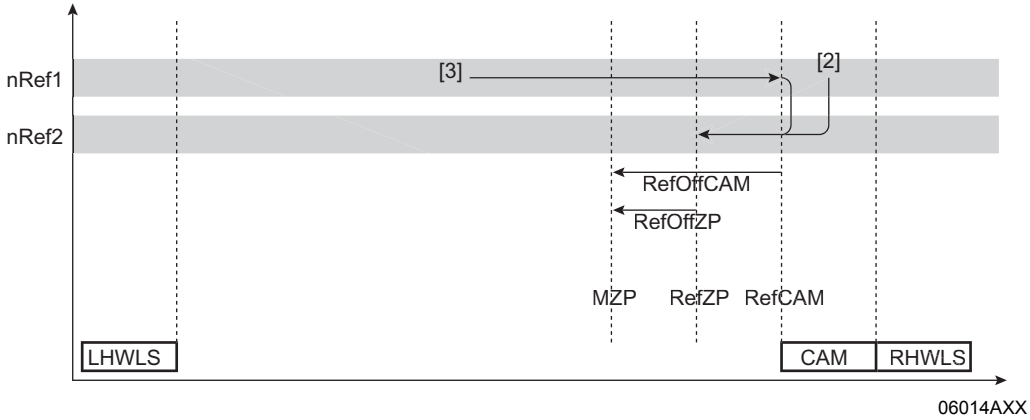


Figura 8: Referenciamento tipo 6

Tipo 7: Came de referência alinhado com a chave fim-de-curso antihorária

A posição de referência é a extremidade direita do came de referência ou o primeiro pulso zero da direita após o final do came de referência.

Uma das entradas digitais P600 ... 606 deve ser ajustada para "REFERENCE CAM." O referenciamento inicia no sentido antihorário; P901 Reference speed 1 é utilizado até o primeiro flanco positivo do came de referência, então é utilizado P902 Reference speed 2. Ao contrário do tipo 2, o conversor inicia no sentido antihorário e gira no came de referência

Se o referenciamento é iniciado através do flanco positivo na entrada "REF.TRAVEL START", o conversor é referenciado na queda de flanco do came de referência ou no pulso zero após a queda de flanco do came de referência, dependendo do ajuste no P904 Referencing to zero pulse

Se o referenciamento é iniciado com o comando IPOS$^{plus®}$ `Go0`, o conversor é referenciado na queda de flanco do came de referência ou no pulso zero após a queda de flanco do came de referência, dependendo se o argumento `"ZP"` ou `"CAM"` é ajustado

O came de referência deve iniciar antes ou de acordo com a chave fim-de-curso antihorária do hardware e deve projetar para a chave fim-de-curso. Isto garante que nenhum contato seja feito com a chave fim-de-curso do hardware durante o referenciamento.

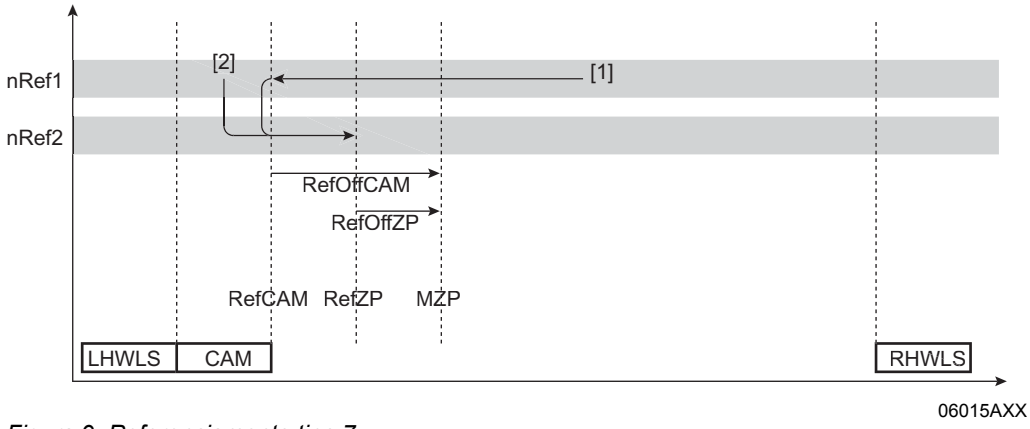


Figura 9: Referenciamento tipo 7

Tipo 8: Sem referenciamento

A posição de referência é a posição atual. Os argumentos no comando IPOS$^{plus®}$ `Go0 "ZP"` ou `"CAM"` e P904 não têm função.

Faz sentido utilizar este tipo de referenciamento com encoders absolutos e para conversores que precisam ser referenciados parados. Por exemplo, a posição de um eixo pode ser ajustada para zero quando o conversor está parado. Deste modo, o operador da máquina pode dizer onde o conversor está localizado dentro de cada movimento.

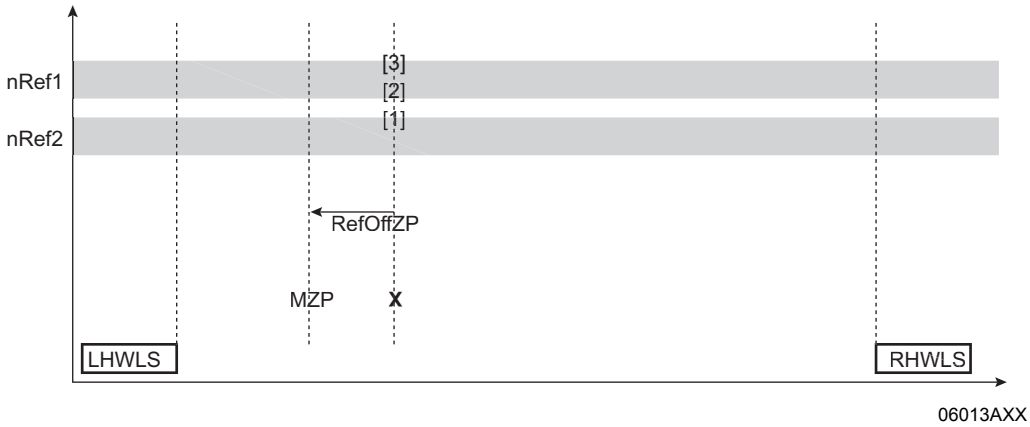


Figura 10: Referenciamento tipo 5

Ao contrário do tipo 5, o referenciamento tipo 8 também pode ser realizado quando o estado do sistema não está ajustado para "A".

5.7 Função Módulo

Introdução

A função módulo pode ser ativada em sistemas giratórios contínuos de um sentido como por exemplo, mesas giratórias ou transportador de correntes. Isto garante que sejam mostrados todos os dados de posição na faixa 0 (valor do módulo – 1 incr.). Quando o valor do módulo definido pelo usuário (por exemplo, 100 mm ou 360°) é excedido, o valor de posição do módulo volta a zero.

Também é possível utilizar o valor incremental do encoder selecionado em P941 para posicionamento e ativar somente a função módulo, por exemplo, para contagem de voltas da saída.

A função módulo tem as seguintes características:

- Especificação da posição nas unidades de saída. Deste modo, uma rotação de 360° da mesa giratória pode ser especificada diretamente sem ter que convertê-la em incrementos do encoder IPOS$^{plus®}$ como no passado. Por exemplo:
  - Rotação de 360° da mesa giratória = valor do módulo = 2^{16} incrementos.
  - Distância abrangida em um ciclo da máquina = valor do módulo = 2^{16} incrementos (mesa giratória com 4 estações: 1 ciclo = 90° = valor do módulo).
- Posicionamento sempre preciso sem desvio a longo prazo ou erros de posicionamento, mesmo para relações de redução não inteiras, contanto que sejam aderidas as orientações do planejamento de projeto.
- Soluções prévias incluso montagem no encoder síncrono externo ou utilizando uma entrada digital para registrar a passagem pelo ponto zero, que significa programação adicional no IPOS$^{plus®}$.
- Especificação da posição absoluta durante várias voltas.
- Especificação de uma estratégia de percurso: O valor nominal de posição pode ser alcançado através da instalação mais curta ou Horário / Antihorário.

Pré-requisitos do hardware e software:

- MOVITOOLS® versão 2.6 e mais atual
- MOVIDRIVE® MDx60 ou MCx4x com avaliação do encoder e processador C168
- MOVIDRIVE® B
- Estado do firmware 823 854 5.10 e maior

O posicionamento contínuo em combinação com a avaliação do encoder absoluto através da DIP11 é possível a partir do firmware versão 14 (822 890 6.14) (irregularidade F92 área de trabalho DIP não ocorre mais, quando a função módulo é ativada).

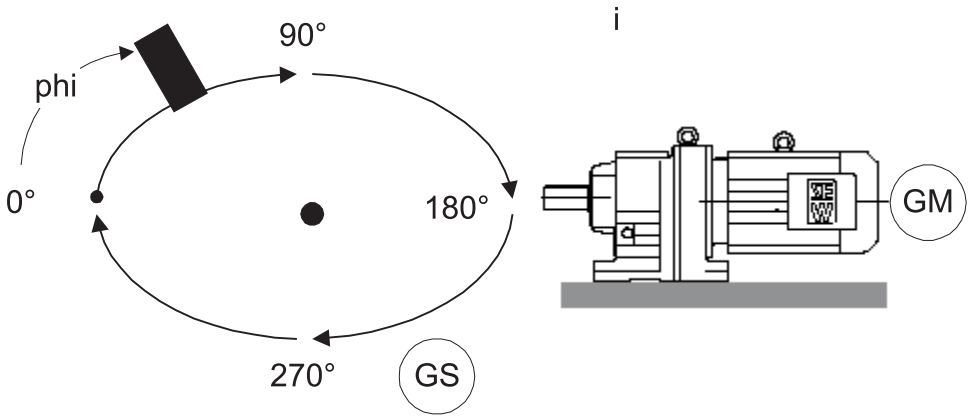
Dependendo do sistema, o redutor e, se necessário, qualquer redução adicional deve ser simulada através do número de dentes. Consultar o fabricante do redutor para o número exato de dentes. Não utilizar a redução da placa de identificação. Além disso, a posição final máxima que pode ser representada é determinada quando o redutor é selecionado. Este valor não deve ser excedido. Isto deve ser considerado na fase do planejamento de projeto (ver a seção Planejamento de Projeto).

Princípio de operação

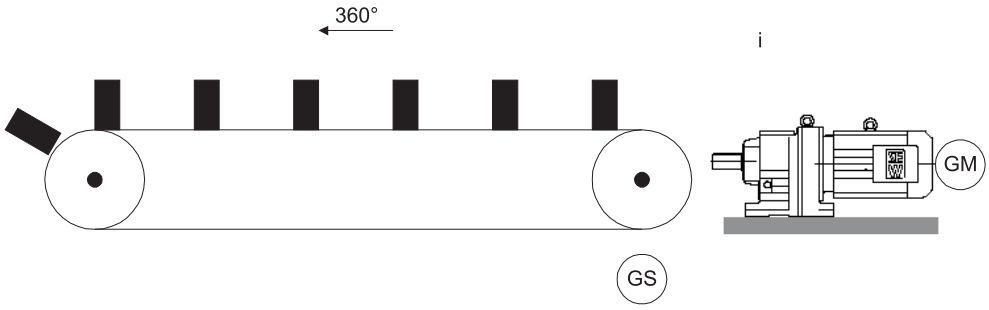
Quando a função módulo está ativa, os valores nominais de posição são expressos nas unidades de saída mais do que nos incrementos no eixo do motor.

Exemplos das unidades de saída (GM = encoder do motor / GS = encoder síncrono):

- Sistemas de mesa giratória com a unidade de saída 360° corresponde a uma volta da mesa giratória.



- Transportador de corrente com a saída 5 poleg. corresponde a um espaçamento da bucha entalhada.



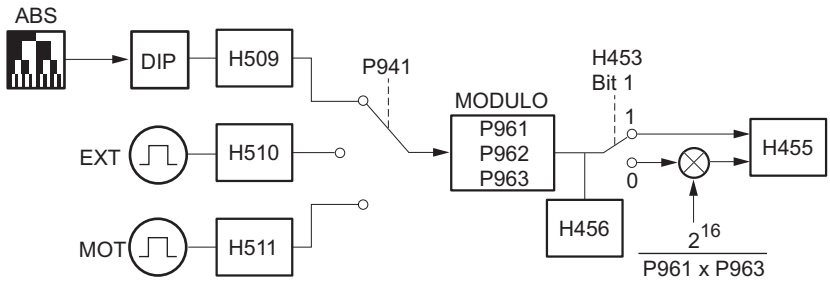
O mecanismo da aplicação é simulado durante a colocação em operação. Para isto, é necessário saber o número exato de dentes no redutor e a redução adicional, se houver. Esta informação é dada nos seguintes parâmetros SHELL:

- Módulo numerador e denominador P961 / P962
- Módulo número de pulsos do encoder P963

O usuário pode utilizar as seguintes variáveis do sistema IPOS$^{plus®}$ para especificar as posições finais nas unidades de saída para o software do sistema MOVIDRIVE®:

- Posição final módulo H454 para descrever a posição final
- Posição atual módulo da saída H455 para ler a posição atual

A posição atual H455 é calculada conforme o seguinte diagrama em blocos do circuito:



P941 = Fonte de posição atual
P961 = Módulo numerador
P962 = Módulo denominador
P963 = Módulo número de pulsos do encoder
H509 = Posição atual de corrente do encoder absoluto
H510 = Posição atual de corrente do encoder externo
H511 = Posição atual de corrente do encoder do motor
H453 = Controle do módulo
H455 = Posição atual do módulo da saída
H456 = Módulo numerador

Se o conversor buscar uma posição de destino dentro da faixa do módulo, ele inicia o posicionamento quando a posição de destino é escrita na variável H454 MOD.TAG-POS. Os comandos GO referem-se ao encoder IPOS$^{plus®}$ e não podem ser utilizados para posicionamento do módulo.

Os exemplos a seguir utilizam o ajuste de 360° do valor do módulo. Também é possível escalonar para um outro valor físico diferente do valor do módulo.

O diagrama abaixo mostra a relação entre a posição atual do encoder IPOS$^{plus®}$, por exemplo, encoder do motor H511 e a posição atual na representação do módulo. A posição atual do módulo sempre movimenta-se dentro da unidade de saída, por exemplo, de 0° (= 0 incrementos) a 360° (= 2^{16} incrementos).

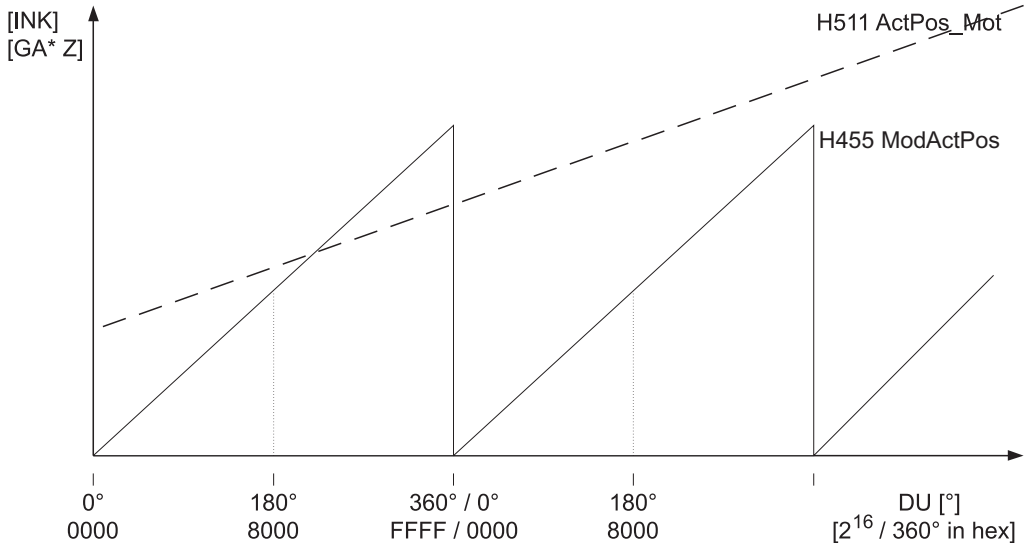


Figura 11: Relação entre as unidades DU [°] e a posição IPOS INK

É especificada uma nova posição final escrevendo a variável IPOS$^{plus®}$ H454 MODTAGPOS no formato 32-bits.

O software do sistema diferencia entre 2 formas de representação, que podem ser ajustadas em H453, bit 1:

- 360° = 16 bits (referido como "2^{16} / 360°") – ajuste padrão: Neste caso, a faixa do bit de maior valor pode ser utilizada para especificar o número inteiro de voltas de 360°.
- 360° = 32 bits (referido como "2^{32} / 360°"): Esta notação deve ser evitada devido a restrição na faixa máxima de representação. Quando utilizar esta forma, o produto do módulo numerador e do módulo número de pulsos do encoder correspondem a uma volta de 360°.

Especificação da posição de amostragem nas unidades de saída (notação hexadecimal):

Representação de várias voltas de números inteiros **H454 MODTAGPOS = k × 360° + 0 ... 360° = k × 2^{16} + 0 ... (2^{16} –1)** **Representação de uma volta de número inteiro** **H454 MODTAGPOS = 0 ... 360°**	
Posição final na unidade de saída [°]	Conversão através da variável IPOS$^{plus®}$ H454 MODTAGPOS
360°	0001 0000
3 × 360°	0003 0000
180°	0000 8000
270°	0000 C000

Estratégias de percurso

Quando a função módulo é ativada, várias estratégias de percurso podem ser utilizadas para posicionamento. A estratégia de percurso para referenciamento depende deste ajuste.

Referenciamento

O referenciamento é iniciado igual ao referenciamento sem a função módulo. Quando a função módulo é ativada, é referenciada a variável H455 MODACTPOS.

Um offset de referência especificado em P900 é interpretado na escala da unidade de saída (2^{16} = 360°).

Uma vez o referenciamento estar completo, a posição final atual H454 MODTAGPOS é ajustada para o valor atual MODACTPOS (ver a seção "Referenciamento").

Posicionamento

A estratégia de percurso para posicionamento é selecionada através do parâmetro SHELL 960 Função Módulo. Os modos podem ser mudados utilizando um programa IPOS$^{plus®}$ com o comando MOVILINK (ver a seção Interface do Usuário). Os exemplos referem-se ao número de pulsos selecionado (2^{16} / 360°).

Um encoder de motor foi inserido como encoder IPOS$^{plus®}$ (P941 Source actual position = MOTOR ENC. (X15)).

"Short route"

Modo "Short route" (P960 = SHORT) – ajuste padrão

Partindo da posição atual de corrente H455 MODACTPOS, o sistema calcula o percurso mais curto para a posição final desejada H454 MODTAGPOS. O sentido de rotação é selecionado com base no percurso mais curto.

Posição final:

H454 MODTAGPOS = k × 360° + 0 ... 360° = k × 2^{16} + 0 ... (2^{16} – 1)

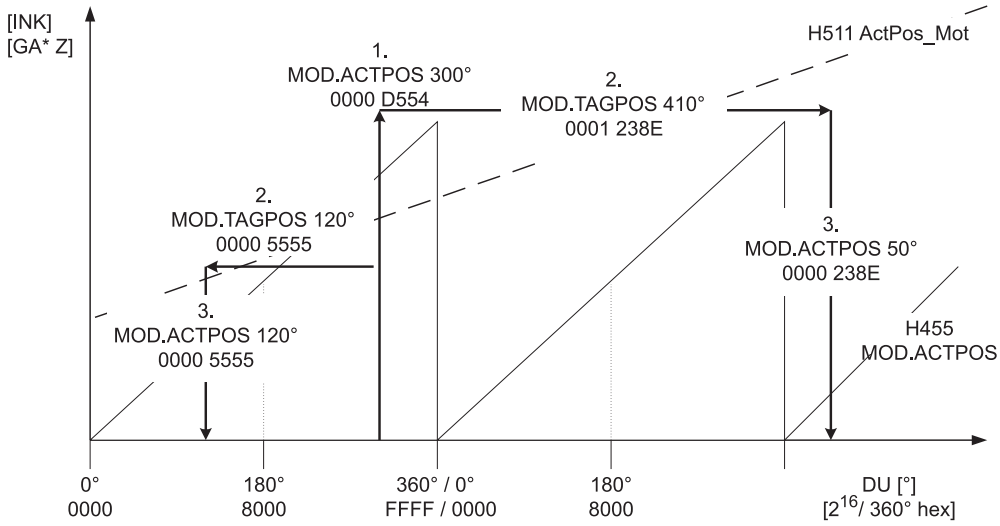


Figura 12: Modo "Short route"

1. Posição atual antes do posicionamento
2. Definição da posição final
3. Posição atual após o posicionamento

Posição atual do módulo
Posição final do módulo
Posição atual do módulo

Módulo short route (observe: com MODTAGPOS = 120° o eixo movimenta-se somente em sentido antihorário quando o conversor está posicionado pelo menos 1 incremento antihorário de 300°, desde que 300° + 180° = 120° e 300° – 180° = 120°).

Portanto, para posicionar um eixo que está em 0° uma volta em sentido horário, H454 deve ser ajustado para 0x 10000. Para movimentar este eixo uma volta em sentido antihorário, H454 deve ser ajustado para 0x FFFF0000.

"CW" Modo "Horário" (P960 = CW)

O conversor movimenta-se da posição atual H455 MOD.ACTPOS horária para a posição final H454 MODTAGPOS.

Posição final:

H454 MOD.TAGPOS = k × 360° + 0 ... 360° = k × 2^{16} + 0 ... (2^{16} – 1)

São permitidos somente valores positivos na parte alta. Se esta condição não for atendida e for ajustado o bit de sinal 2^{32}, o conversor mostra o estado da irregularidade no programa IPOS$^{plus®}$.

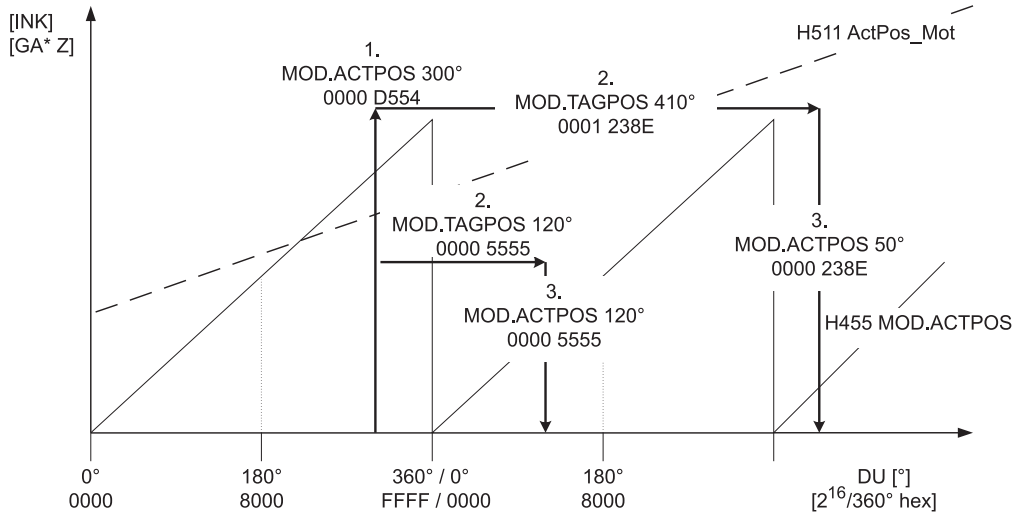


Figura 13: Modo "Horário"

1. Posição atual antes do posicionamento
2. Definição da posição final
3. Posição atual após o posicionamento

Posição atual do módulo
Posição final do módulo
Posição atual do módulo

"CCW"

Modo "Antihorário" (P960=CCW)

O conversor movimenta-se da posição atual H455 MOD.ACTPOS antihorária para a posição final H454 MODTAGPOS.

Posição final:

H454 MOD.TAGPOS = $-k \times 360° + 0 ... 360° = -k \times 2^{16} + 0 ... (2^{16} - 1)$

São permitidos somente valores negativos na parte alta. Se esta condição não for atendida e não for ajustado o bit de sinal 2^{32}, o conversor mostra o estado da irregularidade no programa IPOS$^{plus®}$.

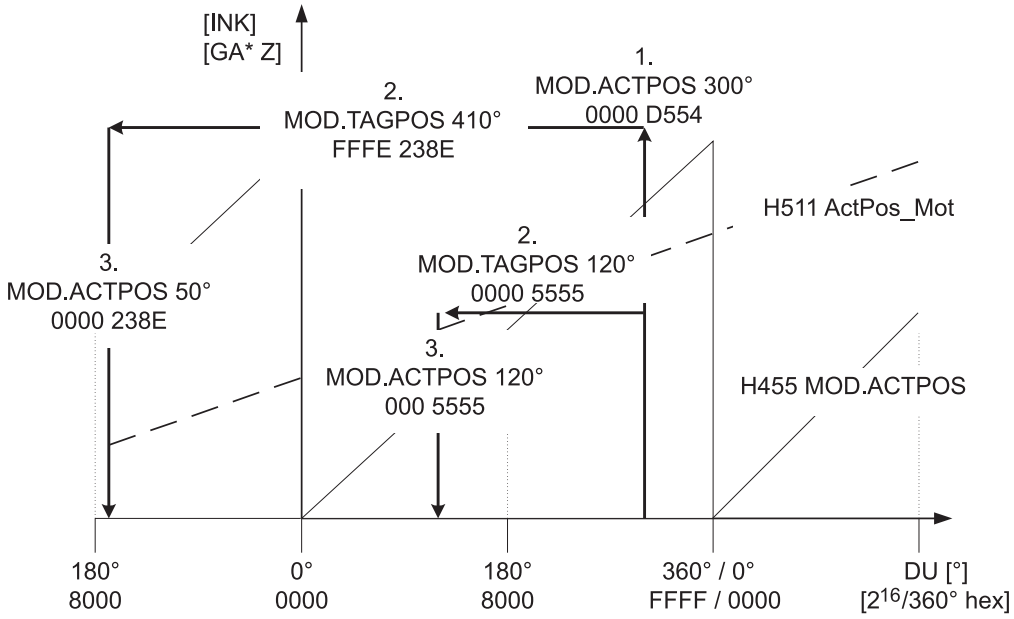


Figura 14: Módulo "Antihorário"

Parâmetros e variáveis SHELL

Parâmetros e variáveis SHELL para a função módulo

Ver as seções "Parâmetros IPOS$^{plus®}$" e "Variáveis do Sistema."

Número do Parâmetro	Nome
P 960, Index 8835	Modulo control
P 961, Index 8836	Modulo numerator
P 962, Index 8837	Modulo denominator
P 963, Index 8838	Modulo encoder resolution

Deve ser realizada a seguinte condição para utilizar a função módulo: O número de pulsos por volta do encoder módulo multiplicado pelo módulo numerador deve ser < 2^{31}.

Número da Variável	Nome
H453	MODULOCTRL
H454	MOD.TAGPOS
H455	MOD.ACTPOS
H456	MODCOUNT

Planejamento de Projeto

Definição da unidade de deslocamento
- O redutor e a redução adicional determinam a unidade de saída 360°
- Determinar a posição final máxima em "Number of drive units"
- Determinar 16 bit ou número de pulsos por volta (encoder × módulo numerador) para 360°

↓

Determinando os parâmetros módulo SHELL
- Modulo function P960 (seleciona a estratégia de percurso)
- Modulo numerator P961
- Modulo denominator P962
- Modulo encoder resolution P963

Recursos para determinar o número de dentes no redutor:
- Manual Técnico SEW
- Programa SEW Wingear para reduzir os fatores numerador/denominador

↓

Faixa de representação e posição máxima de saída do Módulo
- Condição para a faixa de representação: Número de pulsos do módulo encoder × módulo numerador < 2^{31}
- Condição para a posição máxima de saída: = 2^{31} / (número de pulsos módulo encoder × módulo numerador)

Se esta condição não for atendida, pode causar erros no posicionamento!

↓

Implementação no programa IPOS$^{plus®}$
- Especificar a posição do valor nominal com a variável H454 ModTagPos: MOD.TAGPOS = k × 360° + 0 ... 360° = k × 2^{16} + 0 ... (2^{16} – 1)
- Posição atual de leitura na variável H455 ModActPos: MOD.ACTPOS = ... + 0 ... 360° = ... + 0 ... (2^{16} – 1)

O software do sistema lê na posição final especificada no ModActPos e então ajusta a palavra mais significativa para 0.
A posição atual ModuloAbsIst sempre movimenta-se entre 0° e 360°

Exemplos para o planejamento de projeto

Transportador de corrente

Fase 1: Definir a unidade de saída

As posições para um transportador de correntes são especificadas em unidades de saída. Uma rotação de 360° na saída do redutor corresponde a um módulo da unidade de saída de 360°.

Fase 2: Determinar os parâmetros SHELL

Dados técnicos	
Tipo do redutor	KA47B
Rotação de saída [rpm]	19
Rotação do motor [rpm]	2000
Relação de redução do redutor i	104.37
Tipo do motor	DY71S

Os funcionários da SEW podem ler o número de dentes no redutor através do Manual Técnico SEW (DriveNet) ou através da placa de identificação eletrônica (somente para Hiperface®).

Neste exemplo, foram verificados os seguintes números de dentes:

Z1 = 17 / Z2 = 74 / Z3 = 8 / Z4 = 33 / Z5 = 16 / Z6 = 93

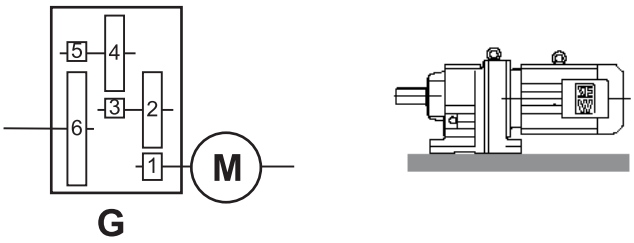


Figura 15: Disposição das engrenagens

Os cálculos a seguir devem ser realizados para determinar os parâmetros SHELL módulo numerador, módulo denominador e módulo de número de pulsos do encoder:

$$\frac{M_N}{M_D} = i_G \times i_{AG}$$

$$\frac{M_N}{M_D} = \frac{Z_2 \times Z_4 \times Z_6}{Z_1 \times Z_3 \times Z_5}$$

$$\frac{M_N}{M_D} = \frac{227106}{2176} = \frac{113553}{1088}$$

M_N = Módulo numerador
M_D = Módulo denominador
i_G = Redução i
i_{AG} = Redução adicional i

O numerador e o denominador foram reduzidos no exemplo acima (acontece automaticamente com o programa Wingear).

Isto resulta nos seguintes valores de entrada para os parâmetros SHELL:

- Módulo numerador = 113553
- Módulo denominador = 1088
- Módulo número de pulsos do encoder= 4096

Fase 3: Faixa de representação e posição máxima de saída do módulo

Verificar a faixa de representação do módulo:

O produto entre o número de pulsos do encoder e o módulo numerador deve ser < 2^{31} (decimal 2147483648).

Módulo numerador × número de pulsos do encoder = 113553 × 4096 = 465113088

=> A condição sendo atendida, a posição final pode ser representada.

Verificar a posição final máxima:

$$TP_{max} = \frac{2^{31}}{M\ 360^\circ} = \frac{2^{31}}{M_N \times M_{ER}} = \frac{2^{31}}{113553 \times 4096} = 4.6$$

TP_{max} =Posição final máxima
M = Módulo
M_N = Módulo numerador
M_{ER} = Módulo número de pulsos do encoder

A posição final máxima corresponde a 4.6 voltas na saída.

Transportador de corrente com bucha entalhada

Fase 1: Definir a unidade de saída

O exemplo acima agora deve ser aumentado:

Uma engrenagem é presa ao redutor, e esta engrenagem aciona uma corrente. Os transportadores do material são colocados a cada 36 elos de corrente. A distância entre os transportadores do material deve corresponder à rotação de 360° da unidade de saída ou 36 polegadas de espaço do transportador.

Fase 2: Determinar os parâmetros SHELL

Dados técnicos	
Tipo do redutor	KA47B
Rotação de saída [rpm]	19
Rotação do motor [rpm]	2000
Relação de redução do redutor i	104.37
Tipo do motor	DY71S

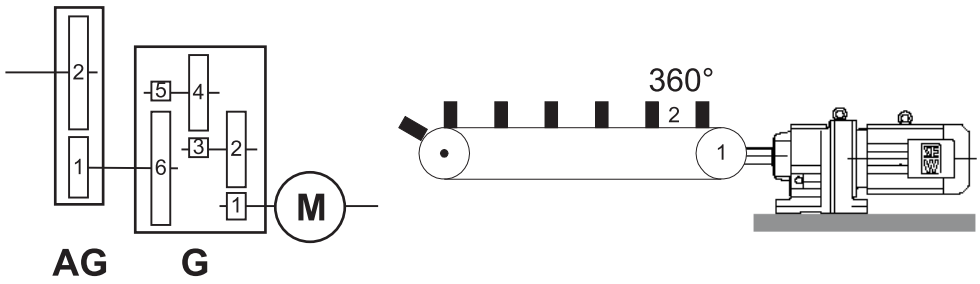


Figura 16: Seleção do redutor

Número de dentes nas correntes para coroa dentada da redução adicional: $Z_{AG}1 = 5$

Espaço do transportador no elo de corrente: $Z_{AG}2 = 36$

Redução adicional i = 5/36

$$\frac{M_N}{M_D} = i_G \times i_{AG}$$

$$\frac{M_N}{M_D} = \frac{Z_2 \times Z_4 \times Z_6}{Z_1 \times Z_3 \times Z_5} \times \frac{Z_{AG}2}{Z_{AG}1}$$

$$\frac{M_N}{M_D} = \frac{1021977}{1360}$$

Módulo numerador = 1021977

Módulo denominador = 1360

Módulo número de pulsos do encoder= 4096

Fase 3: Faixa de representação e posição final máxima do módulo

Verificar a faixa de representação do módulo:

O produto do módulo número de pulsos por volta do encoder e módulo numerador devem ser < 2^{31} (decimal 2 147 483 648).

Módulo numerador $\times$ módulo número de pulsos por volta do encoder = 1021977 $\times$ 4096 = 4 186 017 792

A condição não sendo atendida, a posição final desejada não pode ser representada. Se o usuário implementasse esta aplicação, o resultado seria posicionamento incorreto. O sistema ajusta o bit +/- quando um valor é especificado, por exemplo, uma posição final de 180°; o conversor é posicionado incorretamente.

Solução: Selecionar um redutor com uma redução diferente.

A posição final representada pode ser aumentada selecionando outro redutor com fatores de divisão diferentes para o número de dentes (isto é, parte dos números de dentes cancelaria um outro fora do cálculo).

Questões freqüentes

- Há programas completos para utilizar com a função módulo?
  - Sim, o módulo de aplicação "Módulo Posicionamento" com controle opcional através do fieldbus ou via terminais.
- Por que é recomendado ajustar P960 para SHORT?
  - A estratégia de percurso é aderida estritamente quando a posição final é ajustada durante a parada. Se o conversor é ajustado, por exemplo, para P960 = Horário, quando a posição final é somente um incremento horário, o conversor movimenta-se por uma volta completa.
- Por que o posicionamento não inicia, uma vez a posição final ter sido enviada?
  - O conversor deve ser iniciado no modo de operação "... & IPOS".
  - Uma estratégia de percurso deve ser selecionada através do SHELL P960.
  - Antes de ajustar a variável H454 ModTagPos IPOS$^{plus®}$, o conversor deve estar no estado A "Technology option."
- A posição final pode ser escrita ciclicamente?
  - Para posições de destino na faixa 0° a 359,999° ou 0 incrementos ... 65535 incrementos: Sim.
  - Se as posições finais ≥ 360° são escritas ciclicamente, isto causa "posicionamento contínuo."
- O posicionamento incremental pode ser realizado se a função módulo estiver ativada?
  - Sim, mas para evitar "Efeitos cruzados" indesejados, o módulo posicionamento deve ser desligado no SHELL parâmetro 960.
  - Uma vez o referenciamento estar concluído, somente a variável IPOS$^{plus®}$ H455 ModActPos é ajustada para 0, NÃO a posição atual incremental.
- O que ativa a irregularidade IPOS-ILOOP F10?
  - O intérprete do programa IPOS encontra um comando com operações que não são permitidas.
- Como o posicionamento pode continuar, uma vez a liberação ter sido cancelada?
  - Ajustar o bit H453.0 (ModuloCtrl variable and TargetReset_Off bit).
- Como o eixo age quando o comando _AxisStop(AS_PSTOP) é enviado durante o posicionamento?
  - O conversor espera na rampa de posicionamento; a posição atual é indicada quando o eixo pára. Para continuar o posicionamento, a posição final deve dar um novo valor (por exemplo, mudar o valor do incremento por 1 bit). Uma alternativa seria utilizar o comando _AxisStop(AS_RSTOP) em vez de _AxisStop(AS_PSTOP). Neste caso, a posição final permaneceria inalterada quando o bit H453.0 (ModuloCtrl variable and TargetReset_Off bit) fosse ativado.
- Por que a irregularidade F92 "DIP registration range" é ativada durante posicionamento com o encoder absoluto embora a função módulo está ativa?
  - A irregularidade (fault 92) somente é resolvida no estado firmware .14 (822 890 6.14). Os subconjuntos com um processador 167 (D1) somente podem ser atualizados para versão .11. Eles deveriam ser substituídos por um subconjunto com um processador 168 e estado firmware ≥ .14.

5.8 Controladores de Came Eletrônico

Pode-se utilizar controladores de came eletrônico para ajustar ou reiniciar saídas dependendo da posição de um conversor. Esta função permite controlar atuadores adicionais independente da distância, como cilindros pneumáticos, partida de um segundo eixo ou certificar-se que dois eixos na mesma faixa de operação não colidam.

Cada MOVIDRIVE® vem equipado com um controlador padrão de came eletrônico com uma saída. Uma nova saída é formada toda vez que o comando é processado no programa IPOS$^{plus®}$. Porém, este número de saídas é limitado pelo comprimento do programa IPOS$^{plus®}$, com tempo de processamento aceitável.

Os novos equipamentos MOVIDRIVE® (MDx_A / MCV / MCS / MCF como da versão .14, MCH como da versão .13 e MDx_B) e a versão aplicação têm um controlador de came eletrônico expandido com oito saídas que são calculadas ciclicamente pelo firmware.

O comando GETSYS é utilizado para inicializar um controlador de came eletrônico no conversor e para avaliar o estado dos cames.

O comando GETSYS acessa uma estrutura de dados. O bit de valor mais alto, na primeira variável nesta estrutura de dados, decide qual controlador de came eletrônico o comando GETSYS refere-se (Bit 31 = 0: came eletrônico padrão com bit 31 = 1: came eletrônico expandido).

Se ambos controladores de came eletrônico são disponíveis no equipamento, a SEW recomenda utilizar inicialmente o came eletrônico expandido. Também é possível utilizar ambos controladores de came eletrônico ao mesmo tempo. As saídas de ambos controladores de came eletrônico podem ser colocadas na mesma palavra de saída digital, emitindo as saídas do controlador de came eletrônico padrão nos 2 bits mais baixos e as saídas do controlador de came eletrônico expandido nos 4 bits mais altos, deslocadas duas posições à esquerda.

Se o bit de saída de um came é copiado para uma saída digital do borne, a saída é ajustada 1 ms depois, como com todos os bits que são copiados para saídas no IPOS$^{plus®}$.

Controlador de came eletrônico padrão

Características do controlador de came eletrônico padrão

- É disponível para todos os modos de operação com encoders.
- Por declaração e solicitação de uma estrutura de dados, é ajustada ou reiniciada uma saída com compensação de atraso dependendo das quatro janelas de posição (definida por um valor limite Antihorário e Horário).
- Os limites de uma janela de posicionamento podem ser alterados durante a operação e serão levados em conta com o próximo comando GETSYS. Este opcional possibilita utilizar outras áreas do came para o percurso de retorno no caso de uma reversão do eixo.
- A saída do came pode ser atribuída para qualquer bit de uma variável.
- O número de saídas é limitado pelo comprimento do programa IPOS$^{plus®}$, com tempo de processamento aceitável.
- Será criada uma nova saída com o comando GETSYS, independente se o conversor é referenciado ou não.
- O comando GETSYS inicializa a função e forma o novo estado da saída sempre que o comando é dado. O comando deve ser ativado toda vez que um novo estado é desejado no programa IPOS$^{plus®}$ – a nova geração da saída do came depende da duração de ciclo do programa.
- O valor de referência pode ser ajustado, os tamanhos de referência típicos são:
  - H511 – posição atual do encoder do motor
  - H510 – posição atual do encoder externo
  - H509 – posição atual de um encoder absoluto SSI (opcional DIP11A)
  - H455 – posição atual do encoder do motor no formato módulo
  - H376 – posição atual do valor mestre (somente para versão aplicação came eletrônico ou operação síncrona interna)
- As saídas do came mantém seus valores entre os comandos GETSYS e são apagados somente após um reinício.
- Se a função came é ativada n vezes por 1 ms, podem ser geradas n saídas do came (por ex. em um task rápido, como task 3 no MOVIDRIVE® B, que pode processar vários comandos IPOS$^{plus®}$ por 1 ms). Já que o MOVIDRIVE® gera um novo valor de posição a cada 1 ms, todos os comandos processados durante o período de 1 ms, operam com o mesmo valor de posição.

Iniciando o controlador de came eletrônico padrão

Compiler: _GetSys(Cam1 ,GS_CAM);

Inicializa o controlador de came eletrônico e gera o estado de uma saída com a estrutura de dados igual da variável cam1

Assembler: GETSYS Hxx = CAM

Inicializa o controlador de came eletrônico e gera o estado de uma saída com a estrutura de dados igual da variável Hxx

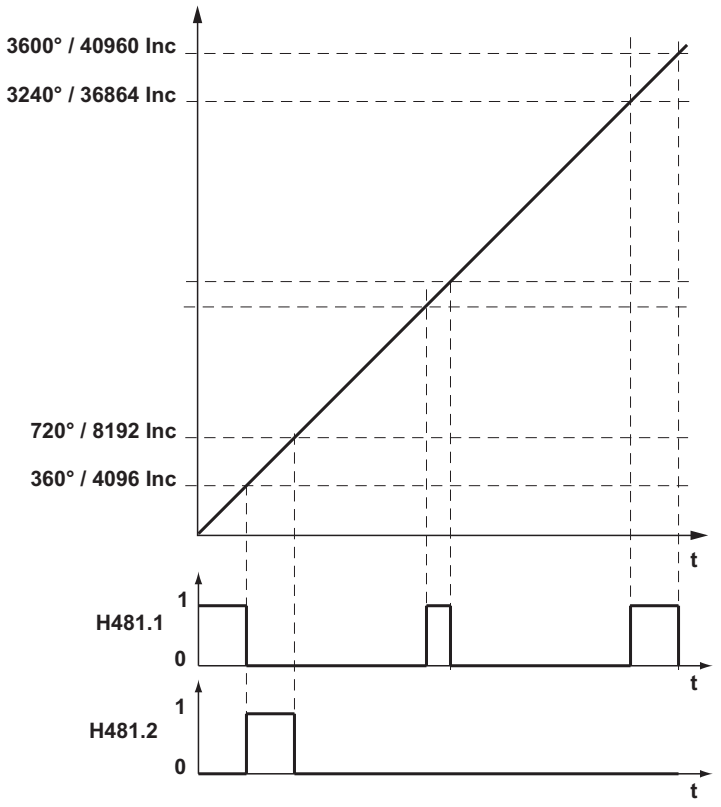
Estrutura de dados do controlador de came eletrônico padrão

Variável	Nome simbólico do elemento na estrutura padrão SEW	Breve descrição
H+0	GSCAM.SourceVar	Número da variável de referência para o cálculo do came, as variáveis de referência típicas são: • H511 (posição atual do encoder do motor) • H510 (posição atual do encoder SSI) • H509 (posição atual do encoder externo) • H455 (posição atual do encoder do motor no formato módulo) por ex. H+0 = 511 para valor de referência H511, o bit 31 da variável deve ser 0.
H+1	GSCAM.DbPreCtrl	Compensação do tempo de atraso em 0.1 ms para compensar o tempo de atraso de um atuador conectado ao conversor. A saída é pré-definida dependendo da taxa de mudança do valor da variável de referência, de tal modo que a saída é comutada por este intervalo de tempo, com antecedência.
H+2	GSCAM.DestVar	Número de variáveis que a saída será ajustada ou reiniciada.
H+3	GSCAM.BitPosition	Posição do bit na variável H+2; se a saída do came é destinada à saída do equipamento (por ex. H481), esta saída digital deve ser reservada com P620 – P639 como uma saída IPOS$^{plus®}$.
H+4	GSCAM.BitValue	Polaridade da saída, 0 = bit ajustado quando a variável de referência H+0 tiver sido ajustada dentro das janelas de posição H+6 a H+13; 1 = bit ajustado quando a variável de referência H+0 estiver fora das janelas de posição H+6 a H+13
H+5	GSCAM.NumOfCam	Número de janelas de posição definido no H+6 a H+13; o valor limite antihorário sempre deve ser menor do que o valor limite horário. Se o módulo eixo necessita uma janela de posição que excede o limite 360° - 0°, então esta faixa tem que ser dividida em duas janelas de posição. Este processo deixa o operador ajustar no máximo três faixas relacionadas a esta saída.
H+6	GSCAM.PosL1	Valor limite antihorário da primeira janela de posição
H+7	GSCAM.PosR1	Valor limite horário da primeira janela de posição
H+8	GSCAM.PosL2	Valor limite antihorário da segunda janela de posição
H+9	GSCAM.PosR2	Valor limite horário da segunda janela de posição
H+10	GSCAM.PosL3	Valor limite antihorário da terceira janela de posição
H+11	GSCAM.PosR3	Valor limite horário da terceira janela de posição
H+12	GSCAM.PosL4	Valor limite antihorário da quarta janela de posição
H+13	GSCAM.PosR4	Valor limite horário da quarta janela de posição

Exemplo

Um transportador tem uma faixa de percurso de dez voltas do motor.

Deve ser ajustada uma saída quando o conversor está na primeira e na última volta do motor ou em uma faixa de +/– 10° perto do centro da faixa de percurso. Uma segunda saída deve ser ajustada quando o conversor está na segunda volta.



Ajustes de parâmetros desejados

P620 = saída IPOS$^{plus®}$

P621 = saída IPOS$^{plus®}$

Exemplo solucionado no compilador

```
/*============================================
IPOS source file
============================================*/
#include <const.h>
#include <io.h>

//Declaration
GSCAM Cam1, Cam2;   //Declaration of cam output 1 and 2

/*============================================
Main function (IPOS initial function)
============================================*/
main ()
{
  /*-----------------------------------
  Initialization
  -----------------------------------*/

  Cam1.SourceVar = numof(ActPos_Mot); // Reference value for motor encoder
  Cam1.DbPreCtrl = 0; // No delay time compensation
  Cam1.DestVar = numof(StdOutpIPOS); // Output on Do01 (H481.1)
  Cam1.BitPosition = 1;
  Cam1.BitValue = 0; // Output = 1, if value in window
  Cam1.NumOfCam = 3; // Number of windows
  Cam1.PosL1 = 0; // 1st window, CCW limit value
  Cam1.PosR1 = 4096; // 1st window, CW limit value
  Cam1.PosL2 = 20366; // 2nd window, CCW limit value
  Cam1.PosR2 = 20594; // 2nd window, CW limit value
  Cam1.PosL3 = 36864; // 3rd window, CCW limit value
  Cam1.PosR3 = 40960; // 3rd window, CW limit value

  Cam2.SourceVar = numof(ActPos_Mot); // Reference value for motor encoder
  Cam2.DbPreCtrl = 0; // No delay time compensation
  Cam2.DestVar = numof(StdOutpIPOS); // Output on Do02 (H481.2)
  Cam2.BitPosition = 2;
  Cam2.BitValue = 0; // Output = 1, if value in window
  Cam2.NumOfCam = 1; // Number of windows
  Cam2.PosL1 = 4096; // 1st window, CCW limit value
  Cam2.PosR1 = 8192; // 1st window, CW limit value

  /*-----------------------------------
  Main program loop
  -----------------------------------*/
   while (1)
  {
    _GetSys(Cam1 ,GS_CAM ); // Create output of 1st cam
    _GetSys(Cam2 ,GS_CAM ); // Create output of 2nd cam
  }

}
```

Exemplo solucionado no assembler

(O comando SET não é necessário e é somente para finalidades demonstrativas)

```
SET     H13   = H511
GETSYS  HO    = CAM
GETSYS  H14   = CAM
END
```

Identifier		Value
HO	Source-Va	13
H1	DbPreCont	0
H2	Dest.-Var	481
H3	Bit-Posit	1
H4	Bit-Polar	0
H5		3
H6		0
H7		4096
H8		20366
H9		20594
H10		36864
H11		40960
H12		0
H13		4016
H14	Source-Va	511
H15	DbPreCont	0
H16	Dest.-Var	481
H17	Bit-Posit	2
H18	Bit-Polar	0

Controlador de came eletrônico ampliado

Características do controlador de came eletrônico ampliado

- Disponível a partir da versão .14 do software MDx_A / MCV / MCS / MCF , MCH a partir da versão .13 do software e MDx_B.
- Pode ser utilizado somente nos modos de operação CFC ou servo com versão aplicação.
- São disponíveis oito saídas (bits do came).
- Pode ser definido para cada saída (corresponde a quatro cames em um disco mecânico do came) até quatro janelas de posição e uma compensação de tempo de atraso.
- As saídas 1-4 são processadas a cada 1 ms; as saídas 5-8 a cada 4 ms.
- O comando GETSYS inicializa e começa a função. Os cames são gerados com um intervalo de tempo fixo e não dependem da duração do ciclo do programa IPOS$^{plus®}$. Fixar valores adequados para as estruturas de dados antes deles serem acessados pela primeira vez.
- São adotadas mudanças para a estrutura de dados a cada 1ms. Esta fase muda os limites de uma janela de posição durante o tempo de realização; estes serão considerados durante o próximo ciclo de processamento do came. Esta opção possibilita utilizar outras áreas do came para o percurso de retorno no caso de uma reversão do eixo.
- Todas as saídas podem ser armazenadas continuamente de qualquer bit de uma variável.
- É possível ajustar as saídas, isto é fixar seus ajustes em 1 ou 0 no programa.
- O valor de referência pode ser ajustado, os tamanhos de referência típicos são:
  - H511 – posição atual do encoder do motor
  - H510 – posição atual do encoder externo
  - H509 – posição atual de um encoder absoluto SSI (opcional DIP11A)
  - H455 – posição atual do encoder do motor no formato módulo
  - H376 – posição atual do valor mestre (somente para versão aplicação came eletrônico ou operação síncrona interna)
- Pode-se parar a função came ampliado acessando GETSYS com bit 31 = 0. Esta fase pára o processamento no firmware e a função não necessita mais qualquer capacidade do processador. Se, no entanto, é atribuído o CamState 0x8000 0000, a função came também será parada mas roda sem ajuste de qualquer saída.

Iniciando o controle do came ampliado

Compiler: _GetSys(CamArray ,GS_CAM);

Inicializa o controlador de came eletrônico e gera o estado de todas as saídas com a estrutura de dados igual da variável CamArray

Assembler: GETSYS Hxx = CAM

Inicializa o controlador de came eletrônico e gera o estado de todas as saídas com a estrutura de dados igual da variável Hxx

Estrutura de dados do controlador de came eletrônico ampliado

A função came eletrônico ampliado é configurada com a ajuda de duas estruturas de dados, isto é, a estrutura CamControl e a estrutura CamOutput.

- A estrutura CamControl controla o comportamento global da função came eletrônico.
- A estrutura CamOutput é utilizada para definir uma saída (saída do disco do came) e pode ser necessária até oito vezes.

Estrutura padrão SEW GSCAM_EXT

Variável	Nome	Descrição
H+0	CamState	Bit 2^{31} deve ser sempre ajustado, caso contrário, o processamento no firmware pára. • 0x8000 000**0** = função inativa, as saídas do came não são mais regeneradas; o ajuste das saídas será retido e apagado somente após um reinício ou quando a tensão está desligada/ligada. • 0x8000 000**1** = função ativa, mas todas as saídas do came serão desligadas. • 0x8000 000**2** = função ativa, se o conversor é referenciado (H473, Bit20 =1) • 0x8000 000**3** = função ativa mesmo sem o conversor referenciado
H+1	CamReserved1	Reservado
H+2	CamOutShiftLeft	Desloca o buffer de dados interno das saídas por n dígitos à esquerda antes de escrever a variável final H+6. Importante: O processo de deslocamento apaga a informação das saídas superiores. Se o fator de deslocamento é 3, as 3 saídas superiores com duração do ciclo de 4 ms não são mais disponíveis, as 4 saídas com duração do ciclo de 1 ms são atribuídas aos bits 3 ... 6 e a saída com duração do ciclo de 4 ms é atribuída ao bit 7.
H+3	CamForceOn	Máscara para forçar saídas obrigatórias. A máscara tem efeito no buffer de dados internos antes do deslocamento com H+2 (NÃO na variável final definida com H+6)
H+4	CamForceOff	Máscara para forçar o apagamento das saídas. A máscara tem efeito no buffer de dados internos antes do deslocamento com H+2 (NÃO na variável final definida com H+6) CamForceOff domina CamForceOn
H+5	CamSource	Bit 2^{31} comuta entre variáveis de referência pré-definidas e um indicador para uma variável de referência aleatória. Bit 2^{31} = 0: • 0 = encoder X15 (encoder do motor, H511) • 1 = encoder X14 (encoder externo, H510) • 2 = encoder H509 (encoder absoluto DIP11A) • 3 = encoder virtual • são reservados todos os valores a seguir! Bit 2^{31} = 1: CamSource inclui um indicador para uma variável IPOS$^{plus®}$ +2^{31}
H+6	CamDestination	Indicador para a variável final. Os bits não utilizados na palavra das variáveis finais são disponíveis para outras funções (se deslocar as saídas quatro posições para a esquerda com Shift Left, estas liberam até os bits 0-3, os bits 4-7 são disponíveis para a função came e os bits 8-31 são disponíveis para qualquer programação.) Se as saídas do came são destinadas às saídas do equipamento (por ex. H481), é preciso reservar estas saídas digitais com P620 – P639 como saídas IPOS. Os bits não utilizados nesta palavra são disponíveis para outras saídas.
H+7	CamOutputs	Número de saídas (máx. 8)
H+8	CamData 1	Indicador para a primeira estrutura CamOutput (primeira saída)
...	...	
H+15	CamData 8	Indicador para a última estrutura CamOutput (oitava saída)

Estrutura CAM_EXT_OUT

Variável	Nome	Descrição
H+0	DeadTime	Compensação do tempo de atraso para este canal (–500 ms..0..+500 ms) para compensar o tempo de atraso de um atuador conectado ao conversor. A saída é pré-definida, dependendo da taxa de mudança do valor da variável de referência, de tal modo que a saída é comutada por este intervalo de tempo, com antecendência .
H+1	CamAreas	Número de janelas de posição para este canal (1 ... 4); o valor limite antihorário sempre deve ser menor do que o valor limite horário. Se um módulo eixo necessita uma janela de posição que excede o limite 360° - 0°, então esta área terá que ser dividida em duas janelas de posição. Este processo deixa o operador ajustar três faixas relacionadas a esta saída.
H+2	LeftLimit1	Limite antihorário, janela 1
H+3	RightLimit1	Limite horário, janela 1
...	...	...
H+8	LeftLimit4	Limite antihorário, janela 4
H+9	RightLimit4	Limite horário, janela 4

Diagrama de função do controle de came eletrônico ampliado

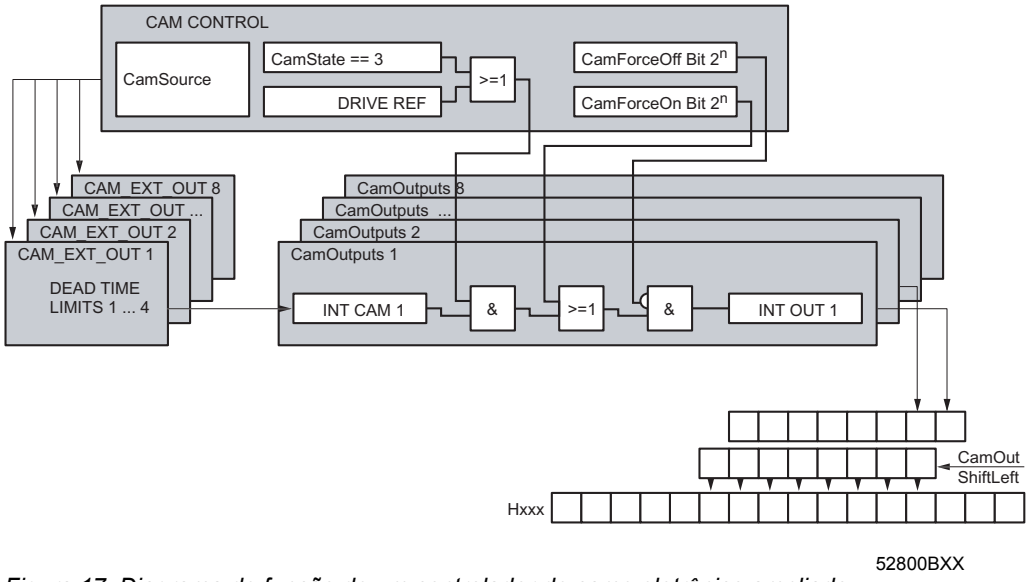


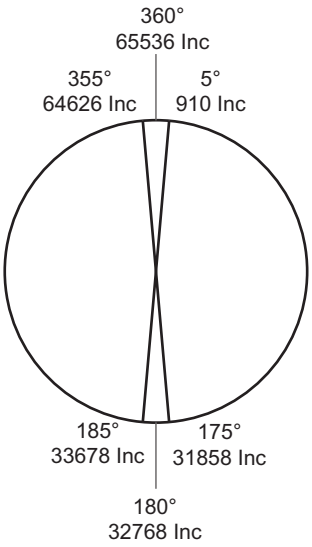
Figura 17: Diagrama de função de um controlador de came eletrônico ampliado

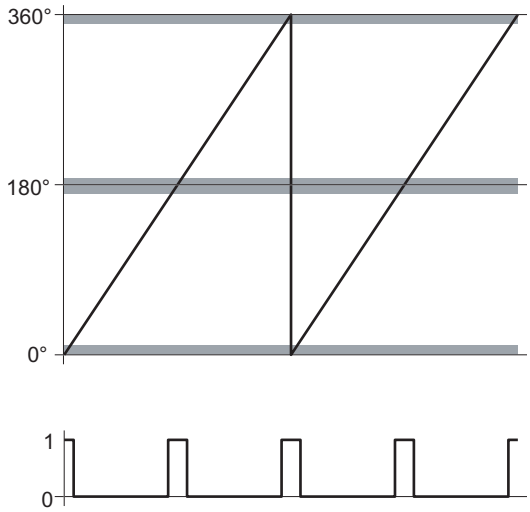
INT CAM 1: Sinal do came interno 1
INT OUT 1: Sinal de saída interno 1
Hxxx: Variável de saída definida com CamDestination

Exemplo

Um módulo mesa giratória tem duas estações de processamento instaladas 180° de distância uma da outra. É acionado por um redutor com uma redução de 1:5. Deve ser ajustada uma saída quando o conversor está na faixa de estações de +/-5°.

(Comentário: uma função módulo ativa efetuará uma rotação de 360° em plena carga com 65536 incrementos – ver função módulo)





Ajustes de parâmetros desejados

P620 = Saída IPOS$^{plus®}$

P960 = por ex. SHORT

P961 = 5

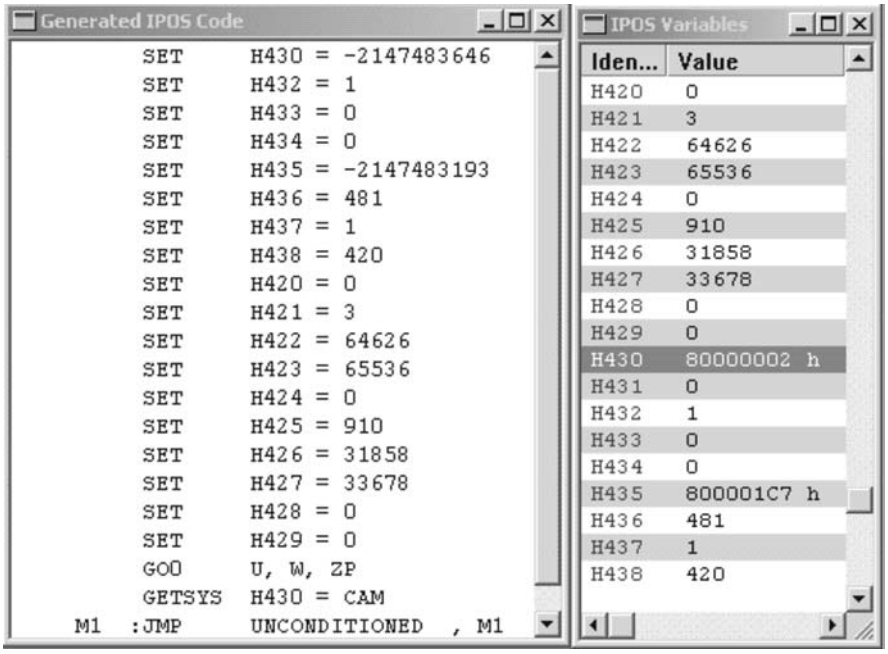
P962 = 1

P963 = 4096

Exemplo solucionado no compilador

```
/*
============================================================================
Program frame for applications with expanded cam function
============================================================================
*/

#include <const.h>
#include <io.h>
// Variable structure for the cam controller is defined externally in *.h
#include <camdef.h>

CamStructure CamData0;
ControlStructure MyCamControl;

/*============================================
Main function (IPOS initial function)
============================================*/
main()
{
  /*-------------------------------------
  Initialization
  -------------------------------------*/
  MyCamControl.CamControl=0x80000002;   // exp. function active
  MyCamControl.CamOutShiftLeft=1;
  MyCamControl.CamForceOn=              // Mask: Mandatory activation of outputs
  MyCamControl.CamForceOn=;           // Mask: Mandatory deactivation of outputs
  MyCamControl.CamSource=numof(ModActPos) | (1<<31); // Actual position value in modulo
format
  MyCamControl.CamDestination=481;      // Basic unit outputs
  MyCamControl.CamOutputs=1;            // Number of cam discs (max. 8)
  MyCamControl.CamDataStr0=numof(CamData0); // Start cam structure 1 (output bit 0)

  CamData0.DeadTime=0;
  CamData0.CamAreas=3;            // 3 cam ranges due to modulo overflow in the window
  CamData0.LeftLimit1= 64626;    // 355° at load = 360° x 64626/65536
  CamData0.RightLimit1= 65536;    // 360° at load
  CamData0.LeftLimit2= 0;        // 0° at load
  CamData0.RightLimit2= 910;      // 5° at load
  CamData0.LeftLimit3= 31858;    // 175° at load
  CamData0.RightLimit3= 33678;    // 185° at load
  CamData0.LeftLimit4= 0;        // Not used
  CamData0.RightLimit4= 0;        // Not used

  _Go0( GO0_U_W_ZP );
  _GetSys( MyCamControl.CamControl ,GS_CAM );
  /*-------------------------------------
  Main program loop
  -------------------------------------*/
  while(1)
  {
  }
}
```

Exemplo solucionado no assembler



6 IPOS$^{plus®}$ e Fieldbus

6.1 Introdução

É utilizado um fieldbus para trocar os sinais de entrada/saída, os dados cíclicos de processo e os parâmetros acíclicos entre um CLP e os conversores. Para descobrir quais fieldbuses são realizados pelo MOVIDRIVE®, consultar o manual de sistema adequado.

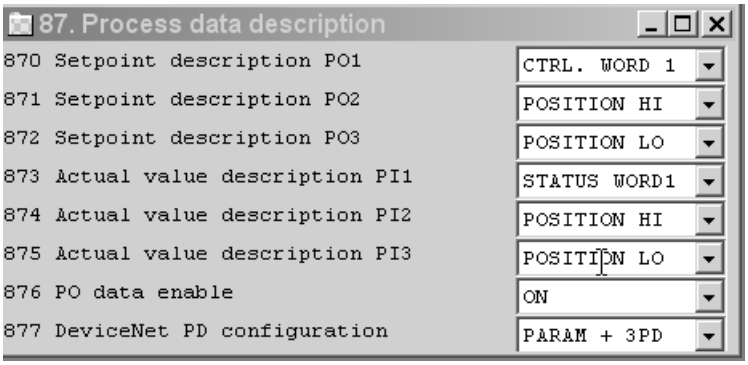
Para uma descrição mais detalhada dos fieldbuses, consultar o manual 'Fieldbus Unit Profile' e os manuais para os fieldbuses individuais. Para controlar o MOVIDRIVE® através do fieldbus, normalmente tem que mudar os seguintes parâmetros:

- P100 Setpoint source = fieldbus, se os valores nominais forem enviados via fieldbus
- P101 Control signal source = fieldbus, se as palavras de controle forem enviadas via fieldbus
- P870-875 Process data configuration: Especifica os dados a ser ampliados via fieldbus

O código do programa IPOS$^{plus®}$ é geralmente o mesmo para fieldbuses; isto é, é idêntico para INTERBUS e PROFIBUS.

Vários exemplos de programa mostrando a conexão entre IPOS$^{plus®}$ e fieldbus estão inclusos no manual "Fieldbus Unit Profile".

Os parâmetros P870 ... P877 podem ser utilizados para ajustar até 3 palavras de dados de processo em ambos sentidos sem um programa IPOS$^{plus®}$. Dependendo do fieldbus utilizado, podem ser trocadas mais palavras de dados de processo (Profibus 1 ... 10 PD, CANOpen 1 ... 8 PD, Interbus LWL 1 ... 6 PD, Ethernet 1 ... 10 PD). Todas as palavras a partir da 4ª palavra são atribuídas a denominação IPOS$^{plus®}$ PI DATA ou IPOS$^{plus®}$ PO DATA. Os dados de processo que recebem estas denominações não são interpretados diretamente pelo conversor. No entanto, todos os dados de processo podem ser acessados através das estruturas de dados do comando GETSYS ou SETSYS.



10337AXX

Figura 18: Seleção dos dados de processo com 3 palavras

6.2 Entradas e saídas digitais

Se nem uma DIO nem uma DIP forem inseridas no MOVIDRIVE®, os bits na palavra de controle 2 / palavra de estado 2 podem ser endereçados no programa IPOS$^{plus®}$, conforme a seguir:

- Diretamente com os nomes simbólicos DI10 ... DI17 ou DO10 ... DO17
- Lido indiretamente com GETSYS e escrito com SETSYS

Neste caso, eles podem ser descritos como entradas e saídas digitais simuladas através de terminais virtuais. O significado dos terminais pode ser ajustado através dos parâmetros P610 ... P617 ou P630...P637.

Interface fieldbus, DIO e DIP

Se for inserida uma DIO ou DIP no MOVIDRIVE®, a organização da seleção dos terminais movimenta-se como descrito para as variáveis afetadas do sistema (ver abaixo), e os dados de processo somente podem ser acessados via GETSYS e SETSYS:

Equipamento	Saídas	Entradas
MOVIDRIVE® A	H481 StdOutpIPOS H480 OptOutpIPOS H482 OutpLevel (somente leitura)	H483 InpLevel
MOVIDRIVE® B	H481 StdOutpIPOS H480 OptOutpIPOS H521 OutpLevel (somente leitura)	H520 InpLevel

6.3 Dados cíclicos de processo

O dado cíclico de processo é lido e escrito num curto tempo de 5 ms.

Dados cíclicos de processo pré-definidos

Se um valor, por exemplo SPEED, é ajustado nos parâmetros P870 ... P875, o item dos dados de processo é ligado diretamente com um valor interno.

Na figura 18 o conversor recebe uma palavra dupla como um valor nominal de posição. O MOVIDRIVE® copia este valor para a variável H499 SetpPosBus e, se o tipo de rampa P916 é ajustado para BUSRAMP, ele utiliza automaticamente este valor como o valor nominal de posição. Se P916 Ramp type = LINEAR, SINE, SQUARED ou JERK LIMITED, o valor nominal pode ser processado no programa do usuário ou copiado diretamente para uma posição final H492 TargetPos ou H454 ModTagPos, utilizando um dos seguintes comandos.

- TargetPos = SetpPosBus; (Compiler)
- SET H492 = H499 (Assembler)

A palavra dupla com o valor atual de posição que é enviado do conversor para o CLP, no exemplo sempre é a posição de H509 ... H511 do encoder selecionado em P941 Source actual position.

Dados cíclicos de processo específicos do usuário

Os usuários têm a opção de escolher a descrição dos dados cíclicos de processo. Para isto, fixar o ajuste de parâmetro dados PO para os dados de saída ou dados PI para os dados de entrada na configuração dos dados de processo. Neste caso, os dados de saída de processo não são avaliados diretamente pelo MOVIDRIVE® mas devem ser copiados para as variáveis IPOS$^{plus®}$ utilizando os comandos GETSYS (dados PO) ou SETSYS (Dados PI). As variáveis são decifradas no programa IPOS$^{plus®}$. Deste modo, o usuário pode, por exemplo, transferir os valores nominais de posição nas unidades do usuário (por exemplo, voltas do motor) multiplicando ou dividindo o valor transmitido pelo fieldbus antes de ser utilizado para posicionamento.

Exemplo

Devem ser transferidos seis itens dos dados de processo com a descrição específica do usuário (P870-877 = IPOS PI DATA ou IPOS PO DATA). Na palavra de saída 2, 3, o CLP transfere o valor nominal de posição para o conversor; na palavra de entrada 3, o conversor envia a posição atual no formato módulo 1/10° (0.0° ... 360.0).

Compiler

```
#include <const.h>
// Data structures for process data
GSPODATA10 tPA;           //Output data (PLC -> drive)
SSPIDATA10 tPE; //Input data (drive -> PLC)

/*============================================
Main function (IPOS initial function)
============================================*/
main()
{
  /*-------------------------------------
  Initialization
  -------------------------------------*/
  // Initialize fieldbus variables for Getsys and Setsys commands
  tPA.BusType = GS_BT_FBUS; //Process data operation via fieldbus interface s.a.
  tPA.Len = tPE.Len = 6; //PD length 6 words

  /*-------------------------------------
  Main program loop
  -------------------------------------*/
  while(1)
  {
    //Read PO data
    _GetSys( tPA, GS_PODATA );
    // Copy double word 2,3 to modulo target position
    ModTagPos = (tPA.PO3 & 0xFFFF) + (tPA.PO2 & 0xFFFF)<<16; //PO2,PO3
    // .....
    //Regenerate process input data and send to PLC
    tPE.PI3 = 3600* ModActPos/ 65536; //Actual position in 1/10 degree at word 3
    _SetSys(SS_PIDATA, tPE.Len); //Send PD
  } //End while (1)
} // End main============================================================
```

Assembler

```
     SET H320  = 3
     SET H332  = 6
     SET H321  = 6
M1   :GETSYS  H320 = PO-DATA
     SET H300  = H324
     AND H300  & FFFF     hex
     SET H301  = H323
     AND H301  & FFFF     hex
     ADD H300  + H301
     SHL H300  << 16
     SET H354  = H300
     SET H300  = 3600
     MUL H300  * H355
     DIV H300  / 65536
     SET H335  = H300
     SETSYS PI DATA         = H332
     JMP UNCONDITIONED  , M1
```

6.4 Comunicação acíclica

Para cada fieldbus, o MOVIDRIVE® realiza a opção leitura e escrita de todos os parâmetros, variáveis, cam disks e o programa IPOS$^{plus®}$ através da comunicação acíclica (também referido como canal de parâmetro ou parâmetro de serviço). Para isto, não é necessário utilizar o programa IPOS$^{plus®}$ ou fazer qualquer ajuste de parâmetro no SHELL (exceção: DeviceNet: P877 DeviceNet PD configuration).

Os dados no conversor são acessados através do endereçamento no índice. Pode-se descobrir o índice de um parâmetro a partir do manual "Fieldbus Unit Profile with List of Parameters" ou pressionando CTRL-F1 no campo de entrada de um parâmetro no SHELL. O índice de uma variável é o número da variável mais 11000 (por exemplo, H34 tem o índice 11034).

Um parâmetro de serviço é processado num curto tempo de 10 ms.

6.5 Características especiais de comunicação através do SBus

Se utilizar o SBus em vez de um fieldbus como fonte de dados para valores de controle e ajuste, a mesma funcionalidade é disponível como para o fieldbus, mas os dados cíclicos de processo são lidos ou escritos em um curto tempo de 1 ms.

Além disso, é possível receber ou enviar mensagens cíclicas ou acíclicas adicionais através de um SBus com um programa IPOS$^{plus®}$. Para mais informações, ver MOVILINK e SCOM nas seções "Funções do Compilador" e "Comandos Assembler."

6.6 Palavras de controle fieldbus e palavras de estado fieldbus

A seguir, um breve resumo da seleção de palavras de controle e estado. Para informação mais detalhada, consultar o manual "Fieldbus Unit Profile with List of Parameters".

Palavra de controle 1

A palavra de controle 1 inclui as funções mais importantes do conversor do bloco de controle básico, assim como os bits para função de setpoint que são gerados no conversor de freqüência MOVIDRIVE® no byte mais significativo.

Definição	Bit	Função
Definição fixa	0	Regulador bloqueado "1" / Liberação "0"
	1	Liberação "1" / Parada rápida "0"
	2	Liberação "1" / Parada "0"
	3	Controle de retenção
	4	Comutação do integrador
	5	Comutação do ajuste de parâmetro
	6	Reset
	7	Reservado
Funções do valor nominal interno	8	Sentido de rotação - potenciômetro do motor
	9	Aceleração do potenciômetro do motor
	10	Desaceleração do potenciômetro do motor
	11	Valor nominal interno fixo
	12	
	13	Comutação do valor nominal fixo
	14	Reservado
	15	

A palavra de controle 1 possibilita controlar o conversor com somente uma palavra de dados de processo de saída na área I/O ou periférica, do dispositivo de automação de nível mais alto, utilizando as funções de valor nominal interno.

A tabela a seguir mostra a seleção do bloco de controle de nível mais alto com as funções de valor nominal interno.

Bit	Funcionalidade	Seleção
8	Sentido de rotação para potenciômetro do motor	0 = Sentido de rotação horário 1 = Sentido de rotação antihorário
9 10	Aumenta o setpoint com a função potenciômetro do motor Reduz o setpoint com a função potenciômetro do motor	10 9 0 0 = sem mudança 1 0 = desaceleração 0 1 = aceleração 1 1 = sem mudança
11 12	Seleção dos valores nominais fixos internos n11 ... n13 ou n21 ... n23	12 11 0 0 = Valor nominal de rotação através da palavra de dados de processo de saída 2 0 1 = Valor nominal interno n11 (n21) 1 0 = Valor nominal interno n12 (n22) 1 1 = Valor nominal interno n13 (n23)
13	Comutação do valor nominal fixo	0 = Podem ser selecionados valores nominais fixos do jogo de parâmetros ativo utilizando bit 11/12 1 = Podem ser ajustados valores nominais fixos de outro jogo de parâmetros utilizando bit 11/12
14	Reservado	Ajustar bits reservados para zero.
15	Reservado	Ajustar bits reservados para zero.

Estas funções de valor nominal interno são ativadas pelo ajuste do parâmetro P100 para valor nominal fixo ou potenciômetro do motor e ajuste da combinação dos bits na palavra de controle 1. Qualquer valor nominal de rotação inserido através de uma palavra de dados de processo de saída SBus não será mais útil.

Função potenciômetro do motor via fieldbus

O valor nominal da função potenciômetro do motor é controlado através da interface fieldbus do mesmo modo como com os bornes de entrada padrão. A rampa de processo que pode ser inserida através de uma palavra de dados de processo de saída adicional não tem efeito na função potenciômetro do motor. Serão utilizados somente os integradores do potenciômetro do motor.

- P150 Rampa de aceleração T3
- P151 Rampa de desaceleração T4

Palavra de controle 2

A palavra de controle 2 contém os bits de função para as funções mais importantes do conversor no bloco de controle básico; os bornes de entrada virtuais na parte mais significativa da palavra. Estes são os bornes de entrada livremente programáveis que não são disponíveis fisicamente devido o hardware faltante (placa opcional). Deste modo, os bornes de entrada são representados nos bornes de entrada virtuais do fieldbus. Cada borne virtual é destinado a um opcional e borne de entrada **fisicamente indisponível**. Sua funcionalidade pode ser programada como desejado.

Definição	Bit	Função
Definição fixa	0	Regulador bloqueado "1" / Liberação "0"
	1	Liberação "1" / Parada rápida "0"
	2	Liberação "1" / Parada "0"
	3	Controle de retenção
	4	Comutação do integrador
	5	Comutação do ajuste de parâmetro
	6	Reset
	7	Timer 2
Bornes de entrada virtuais	8	Borne virtual 1 = P610 / Entrada digital DI10
	9	Borne virtual 2 = P611 / Entrada digital DI11
	10	Borne virtual 3 = P612 / Entrada digital DI12
	11	Borne virtual 4 = P613 / Entrada digital DI13
	12	Borne virtual 5 = P614 / Entrada digital DI14
	13	Borne virtual 6 = P615 / Entrada digital DI15
	14	Borne virtual 7 = P616 / Entrada digital DI16
	15	Borne virtual 8 = P617 / Entrada digital DI17

Se ambas placas opcionais fieldbus e DIO11 forem inseridas no conversor, as entradas DIO11 têm prioridade. Neste caso, as entradas virtuais não são avaliadas.

Palavra de estado 1

A Status Word 1 contém a informação de estado no bloco básico e o *estado da unidade* ou o *número da irregularidade* no byte de nível mais alto. Dependendo do bit de irregularidade, o estado da unidade é indicado para bit de irregu-laridade = 0 ou no caso de irregularidade (bit de irregularidade = 1), o número da irregu-laridade. O bit de irregularidade é reiniciado pela reativação da irregularidade e é indicado o estado atual da unidade. O significado dos números da irregularidade e o estado da unidade são descritos no manual de sistema ou nas instruções de operação do MOVIDRIVE®.

Definição	Bit	Função
Definição fixa	0	Estágio de saída liberado
	1	Conversor pronto
	2	Dados PO liberados
	3	Ajuste do gerador de rampa atual
	4	Ajuste do parâmetro atual
	5	Irregularidade/Aviso
	6	Chave fim-de-curso horária ativa
	7	Chave fim-de-curso antihorária ativa
Estado da unidade / Número da irregularidade	8	**Irregularidade / Aviso?** Bit 5 = 1 → Número da irregularidade: 01 Sobrecorrente 02 ... Bit 5 = 0 → Estado da unidade 0x1 Regulador bloqueado 0x2 ...
	9	
	10	
	11	
	12	
	13	
	14	
	15	

Palavra de estado 2

A palavra de estado 2 contém a informação de estado no bloco de estado básico e os bornes de saída virtuais DO10 ... DO17 no byte de nível mais alto. Programando as funções do borne para os bornes de saída, todos os sinais tradicionais podem ser processados através do sistema fieldbus.

Definição	Bit	Função
Definição fixa	0	Estágio de saída liberado
	1	Conversor pronto
	2	Dados PO liberados
	3	Ajuste do gerador de rampa atual
	4	Ajuste do parâmetro atual
	5	Irregularidade/Aviso
	6	Chave fim-de-curso horária ativa
	7	Chave fim-de-curso antihorária ativa
Bornes de saída virtuais	8	Borne virtual 1 = P630 / Saída digital DO10
	9	Borne virtual 2 = P631 / Saída digital DO11
	10	Borne virtual 3 = P632 / Saída digital DO12
	11	Borne virtual 4 = P633 / Saída digital DO13
	12	Borne virtual 5 = P634 / Saída digital DO14
	13	Borne virtual 6 = P635 / Saída digital DO15
	14	Borne virtual 7 = P636 / Saída digital DO16
	15	Borne virtual 8 = P637 / Saída digital DO17

Se ambas placas opcionais fieldbus e DIO11 forem inseridas no conversor, as entradas DIO11 têm prioridade. Neste caso, as entradas virtuais não são avaliadas.

Palavra de estado 3

A palavra de estado 3 contém a informação de estado no bloco de estado básico e o *estado da unidade* ou o *número da irregularidade* no byte de estado de nível mais alto. Dependendo do bit de irregularidade, o estado da unidade é indicado para bit de irregularidade = 0 ou no caso de irregularidade (bit de irregularidade = 1), o número da irregularidade. O bit de irregularidade é reiniciado pela reativação da irregularidade e é indicado o estado atual da unidade. O significado dos números da irregularidade e o estado da unidade são descritos no manual de sistema ou nas instruções de operação do MOVIDRIVE®.

Definição	Bit	Função
Definição fixa	0	Motor está girando
	1	Conversor pronto
	2	Referência IPOS
	3	IPOS em posição
	4	Freio aliviado
	5	Irregularidade/Aviso
	6	Chave fim-de-curso horária
	7	Chave fim-de-curso antihorária
Estado da unidade / Número da irregularidade	8	**Irregularidade / Aviso?** Bit 3 = 1 → Número da irregularidade: 01 Sobrecorrente 02 ... Bit 3 = 0 → Estado da unidade 0x1 Regulador bloqueado 0x2 ...
	9	
	10	
	11	
	12	
	13	
	14	
	15	

7 IPOS$^{plus®}$ e Movimentos Sincronizados

7.1 Introdução

Pode-se sincronizar vários conversores MOVIDRIVE®, por exemplo, para aplicações com as seguintes funções:

- Eixos mecânicos rodam por vários conversores (por exemplo, pórticos, sistemas de elevação de múltiplas colunas). Observar: Para mais informações sobre o planejamento de projeto neste caso, consultar o manual "MOVIDRIVE® Multi-Motor Drives").
- A rotação ou posição de um eixo escravo é derivada da posição de um eixo mestre (eixo eletrônico, disco de came eletrônico).

O MOVIDRIVE® oferece funções pré-configuradas do hardware e software para aplicações que podem ser ativadas pelo programa do usuário IPOS$^{plus®}$. Estas são descritas nas seções a seguir.

7.2 Sincronização da rotação através da função mestre/escravo

A sincronização da rotação simples é possível através dos parâmetros SHELL P750 e P751 sem a utilização de um programa IPOS$^{plus®}$ ou versão aplicação.

Um exemplo típico de aplicação é a operação sincronizada de 2 transportadores de correias. Para mais informações, consultar o manual de sistema.

7.3 Operação síncrona com placa opcional DRS

Não é necessário um programa IPOS$^{plus®}$ para utilizar a placa de operação síncrona DRS11.

A placa de operação síncrona DRS11x permite que vários eixos sejam operados em ângulo síncrono um em relação ao outro ou com uma relação proporcional ajustável (redução eletrônica). O sistema diferencia entre acionamentos mestre e escravo. O acionamento mestre, utilizado para posicionamento de um ou mais escravos, também pode ser um encoder incremental. O(s) acionamento(s) escravo segue(m) as posições mestre especificadas.

A base para a operação síncrona é a comparação contínua entre as posições mestre e escravo. O sistema determina a diferença da informação de instalação entre o mestre e o escravo e armazena este valor na forma de sinais de encoder incremental em um contador de diferença interno, que não pode ser acessado pelo usuário. Os sinais digitais, como "DRS SLAVE IN POS", "DRS LAG ERROR", "DRS PREWARNING" e "MASTER STANDSTILL" são ajustados dependendo da base desta diferença. Este contador é avaliado dependendo do modo de operação. Na operação síncrona, o contador de diferença interno é utilizado para corrigir qualquer offset angular entre o escravo e o mestre para 0.

A DRS é controlada utilizando as variáveis H473, H476, H477 e H484 (ver seção "Resumo das Variáveis do Sistema").

A seção a seguir descreve como a DRS pode ser endereçada a partir do programa IPOS$^{plus®}$.

Para informação mais detalhada, consultar o manual "Synchronous Operation Card Type DRS11".

Ativando e desativando a função movimento livre

A variável do sistema H476 DRS CTRL pode ser utilizada para ajustar e reativar as duas saídas programáveis da DRS11x.

Pré-requisito

A DRS11x pode ser comutada para o modo movimento livre utilizando IPOS$^{plus®}$ através de um cabo conectado do borne X40:9 (OUTP0) ao X40:1 (modo livre)

Seqüência de comando no assembler

Ajustar OUTP0 e a entrada DRS "Free running": Um LED vermelho acende na placa DRS quando o modo movimento livre é ativado.

```
BSET H476.0 = 1
```

Voltar o contato para a função operação síncrona: O LED vermelho não acende no modo operação síncrona.

```
BCLR H476.0 = 0
```

Escrever BCLR H476.0.

Seqüência de comando no compiler

```
_BitSet( 476, 0 );
_BitClear( 476, 0 );
```

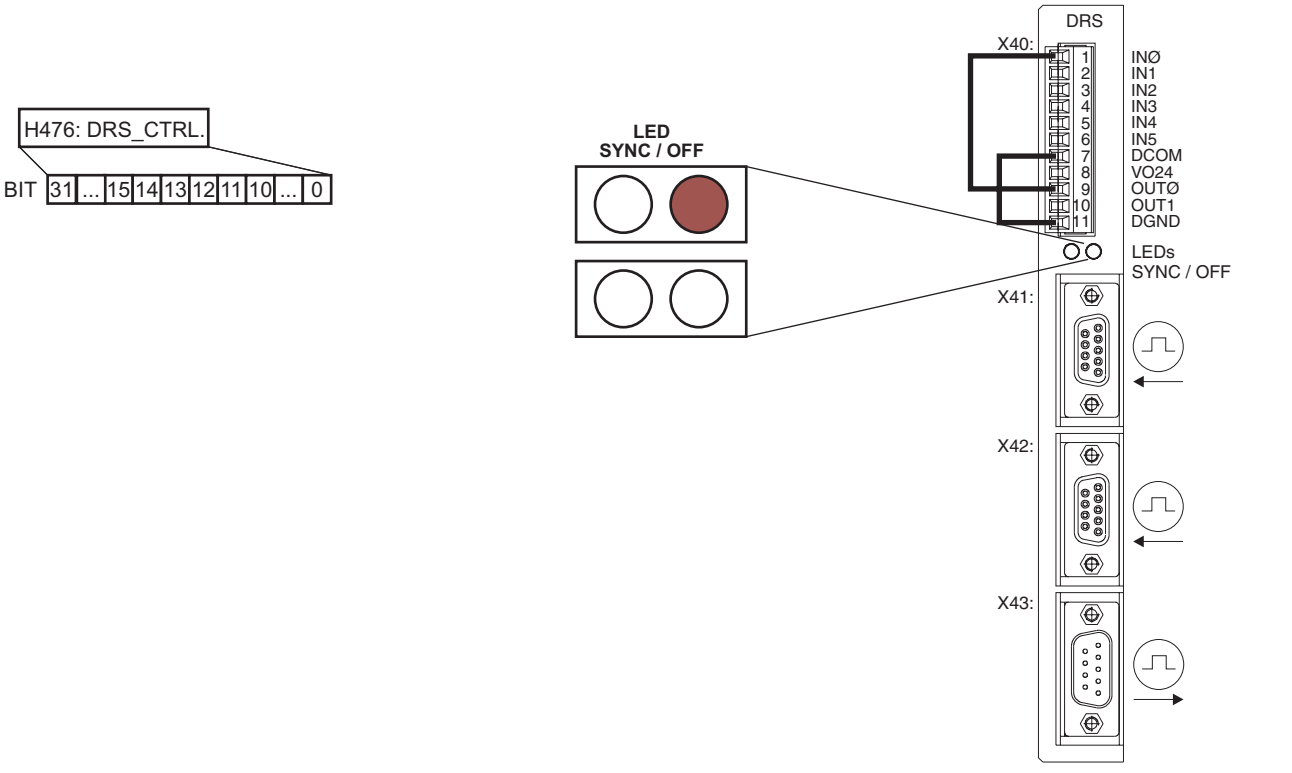


Figura 19: Função movimento livre

Ajustar DRS11A ponto zero

O offset angular da DRS11x pode ser reativado utilizando IPOS$^{plus®}$ sem um sinal externo.

A variável do sistema H484 CTRL. WORD é utilizada para este processo.

Seqüência de comando no assembler

```
BSET H484.22 = 1   Set DRS11A to the 'Set zero point' function
WAIT 15 ms         DRS-specific waiting time of 15 ms
BCLR H484.22 = 0   Reset the 'Set zero point' function
```

Seqüência de comando no compiler

```
 _BitSet( 484,22 );
 _Wait(15);
 _BitClear( 484,22 );
```

O processo de reativar o offset angular entre o mestre e o escravo pode ser monitorado no LED de estado da placa DRS.

Vermelho = movimento livre

Verde = DRS sem posição (offset angular é maior do que P514)

Exemplo de controle

- O conversor deve ser comutado para **Movimento livre** utilizando a entrada **DI10**. As entradas DI10 ... DI17 podem ser bornes físicos na DIO11A e DIP11A ou bornes virtuais na palavra de controle fieldbus 2.
  - DI10 = 1 Movimento livre ativado
  - DI10 = 0 Movimento livre desativado; o conversor roda na operação em sincronismo
- O offset angular atual é apagado através da entrada **DI11** (DRS **Set zero point**).

Exemplo do programa com IPOS$^{plus®}$ Compiler

```
/*====================================================================
IPOS source file
====================================================================*/
#include <const.h>
#include <io.h>
/*----- Define inputs -------------------------------------*/
#define I_FreeRunning         DI10 // Input DI10
#define I_SetZeroPoint  DI11 // Input DI11
/*----- Define outputs ------------------------------------*//
#define O_DRS_AUSG0 0 // Output DRS X40:9
//*----- Define control bits in IPOS control word ---------------*//
#define _DRS_SetZeroPoint 22 // Bit 22

*====================================================================
Subprograms
====================================================================*/
FreeRunning_On()
{
  /* Free running is activated via the external jumper between X40:9 and X40:0
     by setting the output X40:9. */
  _BitSet( DRS_Ctrl, O_DRS_AUSG0 );
}

/*====================================================================*/
FreeRunning_Off ()
{
  /* Free running is deactivated via the external jumper between X40:9 and X40:0
     by deleting the output X40:9. */
  _BitSet( DRS_Ctrl, O_DRS_AUSG0 );
}

/*====================================================================*/
DRS_ZeroPoint()
{
  _BitSet( ControlWord, _DRS_SetZeroPoint ); // Set zero point using control word
  _Wait( 15 ); // Reaction time in ms
  _BitClear( ControlWord, _DRS_SetZeroPoint ); // Delete bit again
}

/*====================================================================
Main function (IPOS initial function)
====================================================================*/
main()
{
  if( I_FreeRunning )            // The input I_free running (here DI10)
    FreeRunning_On();           // switches between
  else                         // free running and synchronous operation
    FreeRunning_Off();

  if( I_SetZeroPoint )     // The function "Set zero point"
    DRS_ZeroPoint();          // is called
}
```

Ativando e desativando a função offset

A variável do sistema DRS CTRL. H476 pode ser utilizada para ajustar e reativar as duas saídas programáveis da DRS11x.

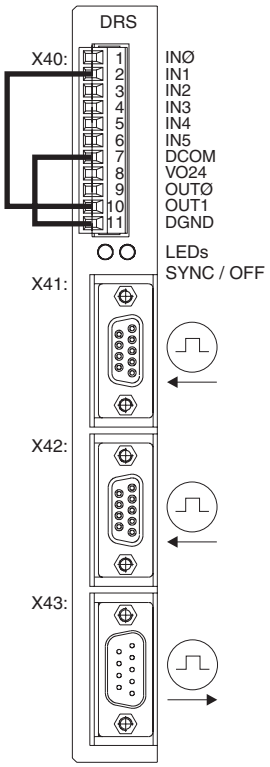


Figura 20: Exemplo de ligação do offset

Pré-requisito

Pode ser aplicado um offset de posição utilizando IPOS$^{plus®}$ através da conexão de um cabo do borne X40:10 ao X40:2 (Offset1).

Seqüência de comando no assembler

Ajustar o conversor para a função "Offset1": O conversor escravo muda sua posição em relação ao mestre para o valor armazenado no offset1.

```
BSET H476.1 = 1
```

Voltar o contato para borne de saída: O conversor escravo volta para a posição anterior em relação ao mestre.

```
BCLR H476.1 = 0
```

Escrever BCLR H476.0.

Seqüência de comando no compiler

```
_BitSet( 476, 1 );
_BitClear( 476, 1 );
```

A saída X40:10 é ajustada utilizando o comando BSET H476.1. É enviado um sinal para X40:2 através de um jumper de X40:10 a X40:2, e a função "Offset" é ativada.

Comutação entre posicionamento e operação em sincronismo

A função "Movimento livre" possibilita que o conversor funcione separadamente em uma rotação controlada. No entanto, quando esta função é ativada, o conversor não pode movimentar-se para uma posição específica utilizando o controle de posição.

Para realizar esta função, é necessário o modo de operação Posicionamento (IPOS$^{plus®}$).

No exemplo do programa a seguir, o sistema pode comutar entre os modos de Operação Síncrona e Posicionamento através do borne de entrada DI12.

Neste contexto, observar a seguinte condição:

O sistema somente pode mudar da operação síncrona para posicionamento nos modos de operação CFC ou SERVO, quando o conversor está em operação.

Para os modos de operação VFC, o MOVIDRIVE® deve estar no estado Regulador bloqueado.

O exemplo do programa a seguir considera uma comutação entre os modos de operação CFC & IPOS e CFC & SYNC com SETSYS.

Aplicam-se os seguintes ajustes (a função borne através do programa IPOS$^{plus®}$ determina os ajustes de parâmetros para todas as entradas utilizadas no IPOS):

- DI10 = 0 Sem movimento livre
- DI10 = 1 Movimento livre, se o modo de operação CFC & SYNC estiver ajustado
- DI11 = 0 Sem função
- DI11 = 1 Ajusta o ponto zero DRS (pulso)
- DI12 = 1 Posicionamento, modo de operação CFC & IPOS
- DI12 = 0 Operação síncrona, modo de operação CFC & SYNC

Os modos de operação podem ser mudados utilizando o comando

```
_SetSys(SS_OPMODE, H)
```

o qual o valor da variável H tem o seguinte significado:

SS_OPMODE: Ajuste do modo de operação
H = 11: Modo de operação CFC (controle de rotação)
H = 12: Modo de operação CFC & controle de torque
H = 13: Modo de operação CFC & IPOS (posicionamento)
H = 14: Modo de operação CFC & operação em sincronismo (DRS11A)
H = 16: Modo de operação SERVO (controle de rotação)
H = 17: Modo de operação SERVO & controle de torque
H = 18: Modo de operação SERVO & IPOS (posicionamento)
H = 19: Modo de operação SERVO & operação em sincronismo (DRS11A)

```
/*==================================================================
IPOS source file
==================================================================*/
#include <const.h>
#include <io.h>

/*----- Define inputs -------------------------------------*/
#define I_FreeRunning         DI10 // Input DI10
#define I_SetZeroPoint DI11 // Input DI11
#define I_Switch_Pos_Sync  DI12 // Input to switch between
// Positioning and synchronous operation
// DI 12 = 1 Positioning  /
// DI 12 = 0 Synchronous operation
/*----- Define outputs ------------------------------------*/
#define O_DRS_AUSG0 0 // Output DRS X40:9
/*----- Define control bits in IPOS control word ---------------*/
#define _FreeRunning 1     // Bit  1
#define _DRS_SetZeroPoint 22 // Bit 22
/*----- Define variables to switch between
/*----- Positioning and synchronous operation ---*/
#define operating mode           H300
#define target position          H0
#define CFC_and_IPOS          13     // Operating mode CFC & IPOS
#define CFC_and_SYNC 14     // Operating mode CFC & synchronous operation

/*==================================================================
Subprograms
==================================================================*/
FreeRunning_On()
{
  /* Free running is activated via the external jumper between X40:9 and X40:0
     by setting the output X40:9. */
  _BitSet( DRS_Ctrl, O_DRS_AUSG0 );
}

/*==================================================================*/
FreeRunning_Off ()
{
  /* Free running is deactivated via the external jumper between X40:9 and X40:0
     by deleting the output X40:9. */
  _BitSet( DRS_Ctrl, O_DRS_AUSG0 );
}

/*==================================================================*/
DRS_ZeroPoint()
{
  _BitSet( ControlWord, _DRS_SetZeroPoint );   // Set zero point using control word
  _Wait( 15 );                                 // Reaction time in ms
  _BitClear( ControlWord, _DRS_SetZeroPoint ); // Delete bit again
}

/*==================================================================*/
Activate_SynchronousOperation()
{
  Operating mode = CFC_and_SYNC
  _SetSys( SS_OPMODE,operating mode );  // Switch operating mode
  DRS_ZeroPoint();                   // Delete angular error
}

/*==================================================================*/
Activate_IPOS()
{
  Operating mode = CFC_and_IPOS;
  _SetSys( SS_OPMODE,operating mode );
}

/*==================================================================
Main function (IPOS initial function)
==================================================================*/
main()
{
  if( I_FreeRunning )
    FreeRunning_On();
  else
    FreeRunning_Off();
  if( I_SetZeroPoint )
    DRS_ZeroPoint();
  if( I_Switch_Pos_Sync )
  {
    Activate_IPOS();
    _GoAbs( GO_NOWAIT,TargetPosition );
  }
  else if ( !I_FreeRunning )
    Activate_SynchronousOperation();
}
```

7.4 Operação síncrona com a versão aplicação "Operação síncrona interna"

A operação síncrona interna é uma solução de firmware utilizada para operar vários eixos em ângulos síncronos. Esta solução do programa utiliza os pulsos de uma unidade mestre. Esta fonte mestre pode ser a entrada X14 (conversor mestre físico) ou qualquer variável IPOS$^{plus®}$ (conversor mestre virtual), por exemplo em conjunto com a SBus ou um encoder virtual. Como na série MOVIDRIVE® B, qualquer fonte pode ser selecionada para a posição atual do eixo (encoder absoluto, encoder síncrono ou qualquer variável IPOS$^{plus®}$).

Deste modo, os eixos com deslocamento também podem ser sincronizados utilizando sincronismo interno. Para MOVIDRIVE® A, somente o encoder do motor pode ser utilizado como a posição atual da fonte.

A operação síncrona compreende várias funções. Uma das funções é que o conversor pode ser posicionado conforme um offset específico e um ciclo para a colocação em operação. Um offset entre o conversor mestre e escravo é ativado após um número específico de incrementos mestre.

Também é realizado o mecanismo de sincronização (processo de sincronização controlado pelo tempo), como visto com a placa de operação síncrona DRS11A. A diferença entre o ângulo do conversor escravo e do conversor mestre resultante do movimento livre é reduzida a zero.

A sincronização também pode ser controlada por posição. O conversor escravo movimenta-se em ângulo síncrono para o conversor mestre seguindo um número específico de incrementos mestre (processo do ciclo para a colocação em operação). Neste tipo de colocação em operação, o conversor escravo movimenta-se com uma rampa quadrática.

Pré-requisitos

A operação síncrona foi projetada para MOVIDRIVE® e coloca os seguintes requisitos no sistema do conversor:

- Realimentação do encoder
- Não são suportadas as placas opcionais DPI11A, DPA11A e DRS
- Modo de operação
  - "CFC" ou "Servo" ... & IPOS
  - "VFC n-control" disponível somente no MOVIDRIVE® B e somente com uma conexão mestre/escravo X14-X14 (estado 07/2003; ainda não disponível através do SBus!)
- As variáveis IPOS$^{plus®}$ H360 a H450 são reservadas para operação síncrona e não devem ser utilizadas no programa de aplicação (ver também seção 3.2 "Resumo das Variáveis do Sistema").
- A operação síncrona é controlada utilizando as variáveis dentro de um programa IPOS$^{plus®}$. Todos os estados da operação síncrona podem ser vistos e ajustados na faixa da variável para operação síncrona de H360 a H450.
- A colocação em operação ISYNC é realizada por uma interface gráfica do usuário.
- O escravo não está sujeito a deslocamento (somente com MOVIDRIVE® A).

Para informação mais detalhada, consultar o manual "Internal Synchronous Operation (ISYNC)".

7.5 Operação síncrona com a versão aplicação "Came Eletrônico"

Um movimento mestre é normalmente representado como um ângulo da máquina entre 0º e 360º. São definidos vários pontos de curva com relação a este ângulo da máquina (o "Plano de movimento"). Estes pontos de controle especificam a posição do conversor escravo particular em relação ao mestre.

O conversor mestre pode ser um conversor físico ou um encoder mestre virtual. O encoder mestre também pode ser comutado utilizando o system bus (SBus) sincronizado. A relação entre as posições do conversor mestre e do conversor escravo é especificada freqüentemente em um gráfico bi-dimensional. A posição do conversor mestre é inserida ao longo do eixo horizontal e a posição do conversor escravo ao longo do eixo vertical. A faixa de posições ao longo do eixo horizontal é referida como ciclo mestre, a faixa de posições ao longo do eixo vertical como ciclo escravo.

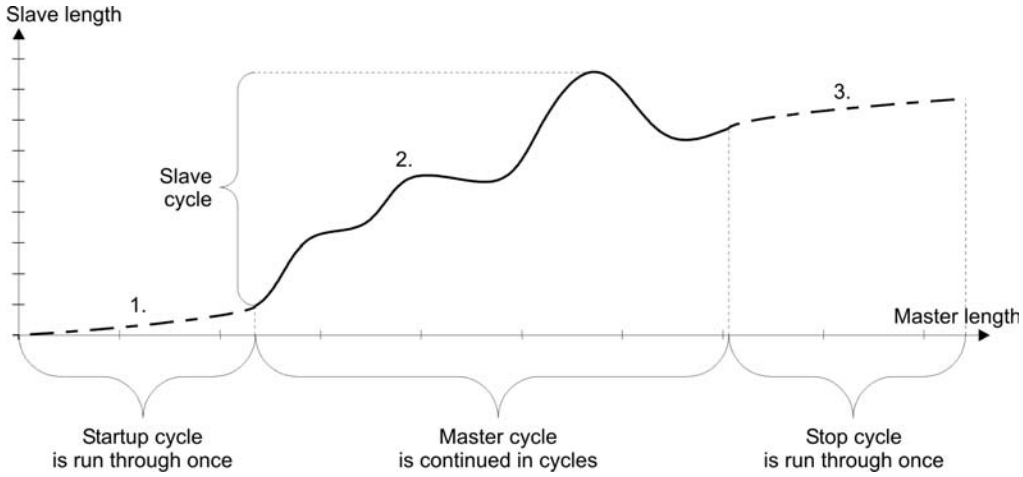


Figura 21: Came eletrônico

[1] Ciclo colocação em operação, funciona completamente uma vez
[2] Ciclo mestre, repetido ciclicamente
[3] Ciclo parado, funciona completamente uma vez
[4] Ciclo escravo
s1 Comprimento mestre
s2 Comprimento escravo

Pré-requisitos

O opcional came eletrônico coloca os seguintes requisitos no sistema do conversor:

- Realimentação do encoder
- Modo de operação: "CFC" ou "Servo" ... & IPOS
- As variáveis IPOS$^{plus®}$ H360 a H450 são reservadas para operação síncrona e não devem ser utilizadas no programa de aplicação (ver também seção 3.2 "Resumo das Variáveis do Sistema").
- A operação síncrona é controlada utilizando as variáveis dentro de um programa IPOS$^{plus®}$. Todos os estados do came eletrônico podem ser vistos e ajustados na faixa da variável para operação síncrona de H370 a H450.

Para informação mais detalhada, consultar o manual de sistema "Addendum to the Electronic Cam".

Se o plano de movimento contém uma inclinação constante, a operação síncrona por fase ocorre como um caso especial no came eletrônico.

8 IPOS$^{plus®}$ para MQx

8.1 Introdução

Da mesma maneira que para módulos MFx, os módulos da série MQx permitem uma conexão fieldbus econômica para conversores MOVIMOT®. Além disso, os módulos MQx são equipados com funções de controle que ajudam a determinar como o conversor responde à entrada externa via fieldbus e I/Os integrados.

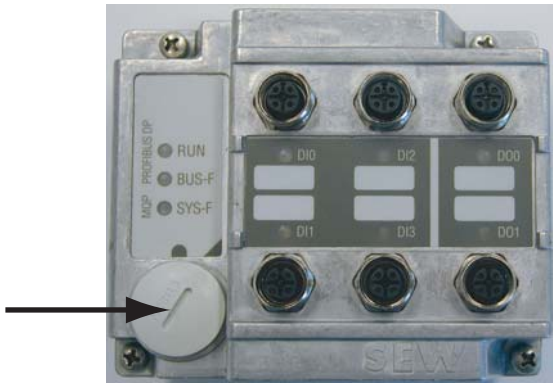
Isto possibilita, por exemplo, que sinais do sensor sejam processados diretamente na interface fieldbus ou permite aos usuários definir seus próprios perfis de comunicação através da interface fieldbus. O sensor de proximidade NV26 permite um posicionamento simples que pode ser integrado em sistemas junto com um programa de controle MQx. Em princípio, o mesmo programa IPOS$^{plus®}$ como utilizado no MOVIDRIVE® A funciona na interface MQx. No entanto, a funcionalidade completa não pode ser realizada em alguns casos. A seção a seguir descreve estas diferenças.

- O grupo de parâmetro P900 Parâmetros IPOS$^{plus®}$ não é executado para a interface MQx.
- Como com o MOVIDRIVE® A, a interface fieldbus MQx fornece dois tasks com os mesmos tempos de processamento para comandos IPOS$^{plus®}$.
- Quando utilizar a variável IPOS$^{plus®}$ H511 (ActPos_Mot), o encoder NV26 não conta 4096 inc./volta mas 24 increm./volta.
- Nem todos os comandos IPOS$^{plus®}$ podem ser utilizados.

8.2 Iniciando a ferramenta de programação

Pode-se acessar a interface fieldbus através da interface de diagnósticos e programação sob um bujão na parte dianteira dos módulos MQx.

Conectar a interface serial do seu PC para o opcional UWS21A e o cabo RJ11 para a interface de programação MQx.



Depois de ter iniciado o MOVITOOLS® Manager e escolhido a interface serial, pressionar a tecla 'Update'. O MQx aparece na janela 'Connected Inverters'. Escolher MQx a partir da janela e iniciar a interface de programação desejada. O compiler permite programação em linguagem de alto nível. O assembler também pode ser utilizado para programação. Os métodos básicos de programação e a faixa de comandos são descritos na ajuda online para a ferramenta selecionada.

8.3 Sistema de controle seqüencial

A seção a seguir indica todas as funções IPOS$^{plus®}$ que podem ser utilizadas no MQx. Pode ser encontrada uma descrição mais detalhada dos comandos nas seções Funções do Compilador e Comandos Assembler. Para informação adicional, consultar

a ajuda online da ferramenta de programação selecionada. Não há limitações comparadas ao MOVIDRIVE® A para rotinas e operadores.

São realizadas as seguintes funções:

_BitClear
_BitMove
_BitMoveNeg
_BitSet
_Copy
_GetSys
_InputCall
_Memorize
_MoviLink
_MovCommDef
_MovCommOn
_Nop
_SetInterrupt
_SetSys
_SetTask2
_TouchProbe
_Wait
_WaitInput
_WdOff
_WdOn

O buffer dos dados de processo fieldbus pode ser acessado através dos comandos _GetSys e _SetSys. A RS-485 para MOVIMOT® pode ser influenciada através dos comandos _MovComm.

8.4 Entradas e saídas digitais

As saídas digitais DO0 e DO1 da MQx (não disponível para MQx32) podem ser ligadas e desligadas utilizando a variável H481 StdOutpIPOS. Para isto, os parâmetros P620 (DO1) e P628 (DO0) devem ser ajustados para Saída IPOS.

H481

...	3	2	1	0
			DO1	DO0

As entradas digitais MQx podem ser lidas utilizando a variável H483 InputLevel. DI4 e DI5 são disponíveis somente com MQx32. Para poder utilizar as entradas como entradas IPOS$^{plus®}$, deve-se ajustá-las para IPOS$^{plus®}$ Input no grupo de parâmetro 62.

H483

...	5	4	3	2	1	0
	(DI5)	(DI4)	DI3	DI2	DI1	DI0

8.5 Avaliação do sensor de proximidade

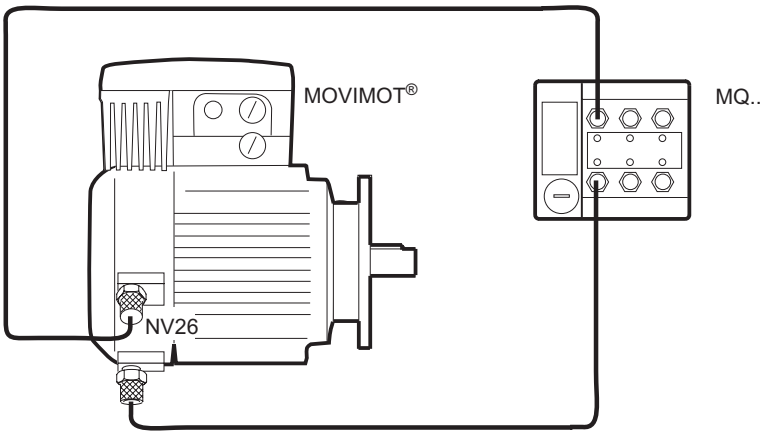
Os módulos MQx permitem posicionamento simples baseado no sistema do sensor de proximidade NV2. Os dois sensores de proximidade enviam 24 informações angulares por volta, que são contadas pelo MQx e armazenadas na variável IPOS$^{plus®}$ H511 (ActPos_Mot) como um valor de posicionamento. Isto permite que o posicionamento seja processado em IPOS$^{plus®}$, que possibilita, por exemplo, que o conversor páre em uma posição específica. Deve ser utilizado uma velocidade reduzida para alcançar a posição exata. O controle de posição não é disponível.

Pode ser conectado somente um MOVIMOT® na interface fieldbus MQx para avaliação do sensor de proximidade. Somente o comando Movcom pode ser utilizado para controlar o MOVIMOT®.

Cabeamento do sensor de proximidade

São necessários para a conexão dois simples cabos blindados de sensor, 4 pólos, com conectores macho e fêmea M12. O cabo conecta os sensores de proximidade NV26 em DI0 e DI1 do módulo de rede.

Recomendamos utilizar conectores macho e fêmea M12 metálicos e conectar a blindagem em ambos os lados.

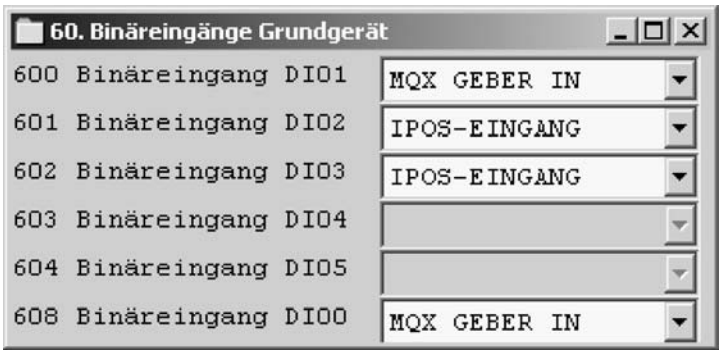


O cabeamento pode ser verificado utilizando os LEDs da MQx. Quando o motor gira lentamente, os LEDs nas entradas DI0 e DI1 piscam.

Se a interface MQx conta a posição do motor em H511 no sentido errado, os conectores M12 nas entradas DI0 e DI1 devem ser ampliados.

Programação dos bornes DI0 e DI1

O sensor de proximidade é avaliado pelo ajuste dos parâmetros para as duas entradas digitais DI0 e DI1 (P600 e P608) em 'MQX ENCODER IN.' Os ajustes dos parâmetros podem ser mudados utilizando a interface MOVITOOLS® SHELL ou no início da seção do IPOS$^{plus®}$. O IPOS$^{plus®}$ pode endereçar qualquer parâmetro MQx no endereço 253 utilizando o comando Movilink. Os índices das entradas são 8844_{dec} para DI0 (P608) e 8335_{dec} para DI1 (P600). São utilizados dois acessos de escrita para transferir o valor 32 para os dois índices. As entradas são filtradas conforme o ajuste de fábrica com 4 ms. A programação dos bornes 'MQX ENCODER IN' comuta este filtro para o desligamento da avaliação do sensor de proximidade.



Monitoração do encoder

Os dois cabos do encoder são verificados contra rupturas do fio. Os cabos são monitorados somente quando o MOVIMOT® está liberado. Se o sinal não muda para 1 s em pelo menos um dos cabos do encoder, é detectada uma ruptura do fio e indicada pela irregu-laridade 14. O MOVIMOT® pára e pode ser iniciado novamente, somente quando o MQx tiver sido reativado. Durante este processo, a posição atual é perdida e o sistema deve ser novamente referenciado. A monitoração do encoder pode ser ligada ou desligada utilizando o parâmetro P504 (Monitoração do encoder para motor).

Armazenagem da posição atual

Quando o comando MOVILINK entra em contato com o endereço 253, a posição atual pode ser armazenada a qualquer hora em uma variável na faixa H0 ... H127 na memória não volátil. Recomendamos salvar a posição cada vez que o conversor movimenta-se para uma nova posição atual. Isto assegura não ter que realizar referenciamento quando reiniciar a unidade. O referenciamento é, no entanto, ainda necessário se a tensão MQx estiver desconectada durante a operação de posicionamento. O módulo é estruturado para suportar 10 bilhões de ciclos de escrita.

8.6 Contador

Os módulos MQx possuem um contador que pode ser conectado na DI0 ou DI1. Os flancos positivos são contados até a freqüência máxima de entrada de 4 kHz. Para ativar a função contador, deve-se mudar o ajuste da entrada correspondente para 'MQX ENCODER IN'. Este ajuste desliga o filtro de entrada automaticamente. Como resultado, o sinal de entrada para o contador deve ser livre de irregularidade e livre de salto. O valor do contador é escrito para a variável H511.

Se as entradas DI0 e DI1 são ajustadas para 'MQX ENCODER IN', a avaliação do sensor de proximidade é ativada automaticamente e a função contador é desligada.

8.7 Valores da variável DIAG11 para a irregularidade IPOS ILLOP

A irregularidade IPOS ILLOP é gerada como uma mensagem de irregularidade geral quando ocorre um erro no programa IPOS$^{plus®}$.

É enviado o número da irregularidade interna para a variável de diagnóstico H469 (DIAG11) para análise detalhada da irregularidade. Toda irregularidade que ocorrer é atribuído seu próprio número de irregularidade. 999 significa que a função ou o argumento da função não é realizado pelo módulo MQx.

Quando utilizar os comandos MOVLNK e MOVCOM podem ocorrer os números de irregularidade a seguir.

DIAG11	Comando	Causa da irregularidade
500	MOVLNK	O comando MOVLNK foi acessado após a comunicação cíclica ter iniciado. O MOVLINK é travado pelo comando _MovCommOn.
501	MOVLNK	Número da variável H, de onde o dado de leitura é armazenado ou de onde é obtido o dado a ser escrito, não encontra-se na faixa válida (H0 ... H450).
502	MOVLNK	O tipo de rede é inválido. Permitido somente 2 = RS-485#2.
503	MOVLNK	Foi inserido um tipo PDU inválido no formato. São permitidas somente carcaças acíclicas (128 ... 134).
504	MOVCOM	O comando _MovCommDef foi acessado após a comunicação cíclica ter iniciado. O _MovCommDef é travado pelo MovComOn.
505	MOVCOM	O _MovCommDef foi acessado pela nona vez. São permitidas 8 relações de comunicação.
506	MOVCOM	O tipo de rede é inválido. permitido somente 2 = RS-485#2.
507	MOVCOM	Foi inserido um endereço inválido da estação. Não são permitidos os endereços 253 e 254 (0 -252, 255).
508	MOVCOM	Foi inserido um tipo PDU inválido no formato. São permitidas somente carcaças cíclicas (0-6).
509	MOVCOM	Número da variável H, de onde o dado de leitura é armazenado ou de onde é obtido o dado a ser escrito, não encontra-se na faixa válida (H0 ... H450).
511	MOVCOM	A estação não suporta o tipo PDU inserido no formato.

9 Parâmetros IPOS$^{plus®}$

9.1 Introdução

Os parâmetros IPOS$^{plus®}$ são descritos detalhadamente no manual IPOS$^{plus®}$.

Perigo de movimentos de percurso inesperados

Possíveis conseqüências: Ferimentos graves ou fatais.

Observe que a modificação destes parâmetros sem conhecimento do programa IPOS$^{plus®}$ que pode estar ativo, pode causar movimentos inesperados e colocar cargas indesejáveis no acionamento mecânico. É necessário estar familiarizado com o manual IPOS$^{plus®}$ para fazer o ajuste destes parâmetros.

9.2 P90x Referenciamento IPOS

O referenciamento é utilizado para estabelecer o **zero da máquina** o qual referem-se todos os comandos de posicionamento absoluto. Pode-se selecionar diferentes estratégias de referenciamento em P903 Tipo de referenciamento. Estas estratégias definem os modos adequados de percurso, por exemplo procurar um came de referência. Baseado no **ponto de referência** estabelecido através do referenciamento, pode-se utilizar P900 Reference offset para deslocar o zero da máquina conforme a equação

Zero da máquina = posição de referência + offset de referência

.

As rotações para os movimentos de percurso necessários conforme o **tipo de referenciamento** são ajustadas com P901 Reference speed 1 / P902 Reference speed 2 .

P900 Reference offset

Faixa de ajuste: $-(2^{31}-1)$ ... 0 ... $2^{31}-1$

O offset de referência (offset zero) é utilizado para determinar o zero da máquina sem movimentar fisicamente a posição de referência. Aplica-se o seguinte:

Zero da máquina = posição de referência + offset de referência

O offset de referência refere-se ao ajuste do encoder via P941 Source actual position quando a função módulo é desativada (P960 = OFF).

Se a função módulo é ativada, o offset de referência refere-se ao ModActPos H455. No assembler, o offset de referência refere-se às unidades de deslocamento do usuário.

- H509 Actual position SSI absolute encoder (SIP)
- H510 Actual position external encoder
- H511 Actual position motor encoder

O offset de referência é novamente ativado após o referenciamento ter sido completado com sucesso.

P900 é idêntico ao H498.

P901 Reference speed 1

Faixa de ajuste: 0 ... 200 ... 6000 rpm

A rotação de referência 1 determina a velocidade de deslocamento para a primeira parte do referenciamento com o came de referência. A mudança de velocidade sempre ocorre através da rampa de parada t13 (P136). Os sentidos de busca durante o referenciamento são determinados pelo respectivo tipo de referenciamento. A velocidade está ativa até o came de referência ter sido alcançado.

P902 Reference speed 2

Faixa de ajuste: 0 ... 50 ... 6000 rpm

A rotação de referência 2 determina a velocidade de deslocamento para o referenciamento no pulso zero (tipo 0) ou para a segunda parte do referenciamento com cames. A mudança de velocidade sempre ocorre através da rampa de parada t13 (P136). Os sentidos de busca durante o referenciamento são determinados pelo respectivo tipo de referenciamento. A velocidade estará ativa até o came de referência ser detectado, até o conversor deixar o came de referência ou até o 1º pulso zero ter sido alcançado.

P903 Reference travel type

O tipo de referenciamento especifica a estratégia de referenciamento que é utilizada para estabelecer o zero da máquina.

Este ajuste também define o sentido de busca para o came de referência nas fases de referenciamento individuais.

Utilizar o parâmetro P904 Reference travel to zero pulse para determinar se o referenciamento ocorre em uma mudança de flanco do came de referência ou no próximo pulso zero do encoder.

Também há tipos disponíveis que podem funcionar sem um came de referência. Os diagramas de percurso podem ser encontrados na seção "Detecção da Posição e Posicionamento".

- **Tipo 0: Pulso zero antihorário**
  - O primeiro sentido de busca é antihorário
  - Posição de referência = pulso zero antihorário a partir da posição atual
  - Zero da máquina = posição de referência + offset de referência
- **Tipo 1: Extremidade antihorária do came de referência**
  - O primeiro sentido de busca é antihorário
  - Posição de referência = Primeiro pulso zero ou queda de flanco à esquerda do came de referência
  - Zero da máquina = posição de referência + offset de referência
- **Tipo 2: Extremidade horária do came de referência**
  - O primeiro sentido de busca é horário
  - Posição de referência = Primeiro pulso zero ou queda de flanco à direita do came de referência
  - Zero da máquina = posição de referência + offset de referência
- **Tipo 3: Chave fim-de-curso horária**
  - O primeiro sentido de busca é horário
  - Posição de referência = Primeiro pulso zero ou queda de flanco à esquerda da chave fim-de-curso horária.
  - Zero da máquina = posição de referência + offset de referência
  - O referenciamento deve ocorrer no pulso zero.
- **Type 4: Chave fim-de-curso antihorária**
  - O primeiro sentido de busca é antihorário
  - Posição de referência = Primeiro pulso zero ou queda de flanco à direita da chave fim-de-curso antihorária.
  - Zero da máquina = posição de referência + offset de referência
  - O referenciamento deve ocorrer no pulso zero, como o conversor pode estar localizado na histerese de contato da chave fim-de-curso após o referenciamento e a irregularidade 29 "Limit switch reached" pode ocorrer esporadicamente.

- **Tipo 5: Sem referenciamento**
  - Posição de referência = posição atual
  - Zero da máquina = offset de referência
- **Tipo 6: Came de referência alinhado com a chave fim-de-curso horária**
  - O primeiro sentido de busca é horário
  - Posição de referência = Primeiro pulso zero ou queda de flanco à esquerda do came de referência
  - Zero da máquina = posição de referência + offset de referência
  - OBS.: O came de referência e as chaves fim-de-curso devem estar alinhados!
- **Tipo 7: Came de referência alinhado com a chave fim-de-curso antihorária**
  - O primeiro sentido de busca é antihorário
  - Posição de referência = Primeiro pulso zero ou queda de flanco à direita do came de referência
  - Zero da máquina = posição de referência + offset de referência
  - OBS.: O came de referência e as chaves fim-de-curso devem estar alinhados!
- **Tipo 8: Sem referenciamento**
  - Posição de referência = posição atual
  - Zero da máquina = offset de referência

  Ao contrário do tipo 5, o referenciamento tipo 8 também pode ser realizado quando o estado do sistema não estiver ajustado para "A".

P904 Reference travel to zero pulse

Faixa de ajuste: SIM / NÃO

O parâmetro não tem efeito se o comando IPOS$^{plus®}$ GO0 é utilizado para referência. É utilizado somente quando o referenciamento é iniciado através da entrada digital REF.TRAVEL START.

- Sim: O referenciamento ocorre no pulso zero do encoder IPOS$^{plus®}$ selecionado.
- Não: O referenciamento ocorre na queda de flanco do came de referência.

P905 Hiperface® offset X15

Faixa de ajuste: $-(2^{31}-1)$ ... 0 ... $2^{31}-1$

Este parâmetro é utilizado para especificar o ponto zero do indicador do encoder do motor.

Este parâmetro é utilizado para definir o zero da máquina do encoder do motor H511 sem referenciamento. Isto adiciona ou subtrai o offset do valor do encoder.

- P905 Hiperface® offset X15 tem efeito na posição atual do encoder do motor H511.

  H511 = Valor do encoder – P905

A posição atual é determinada diretamente após os valores terem sido inseridos. Isto não necessita referenciamento anterior. P947 Hiperface® Offset (X14) trabalha da mesma maneira para o encoder externo.

Observe:

Quando o referenciamento de um sistema de conversor ocorre com um encoder Hiperface®, dependendo do ajuste da fonte de posição atual, o referenciamento irá recalcular e regravar P905 Hiperface® offset X15.

Aplica-se o seguinte:

- P905 = valor do encoder – P900

Observe:

O offset Hiperface® não pode ser utilizado para referência do valor de posição atual ModActPos H455 do módulo de eixo. Neste caso, P903 Reference travel type deve ser ajustado para 5.

9.3 P91_ Parâmetros de deslocamento IPOS

P910 Gain X controller

Valor para o controlador P da rotina do controle de posição no IPOS$^{plus®}$. Quando o MOVITOOLS® é iniciado, os valores para os parâmetros P210 (Ganho P controlador de retenção) e P910 são calculados dos valores de entrada e ajustados para o mesmo valor. Depois, eles podem ser mudados separadamente.

Faixa de ajuste: 0 ... **0,5** ... 32

P911/912 Positioning ramp 1/2

Ajuste do valor para a rampa utilizada durante a operação de posicionamento. Sempre é utilizada a mesma rampa (rampa de posicionamento 1) para aceleração e desaceleração do P916 Ramp type SINE and SQUARED. Com um tipo de rampa LINEAR, a desaceleração é ajustada dependendo do P917 Ramp mode:

O tempo de rampa inserido refere-se à mudança de velocidade de 3000/min.

Faixa de ajuste: 0.01 s ... **1.0 s** ... 20.0 s

P913 Travel speed CW

Velocidade para posicionamento horário. A velocidade de deslocamento é inserida em voltas/min e não deve exceder a velocidade máxima do motor.

Faixa de ajuste: 0/min ... **1500/min** ... 5000/min

O parâmetro 913 sempre deve ser ajustado 10 % inferior ao parâmetro 302 (rotação máxima 1) para evitar erros por atraso (reserva de posicionamento para o controlador de posição)!

P914 Travel speed CCW

Velocidade para posicionamento antihorário. A velocidade de deslocamento é inserida em voltas/min e não deve exceder a velocidade máxima do motor.

Faixa de ajuste: 0/min ... **1500/min** ... 5000/min

O parâmetro 914 sempre deve ser ajustado 10 % inferior ao parâmetro 302 (rotação máxima 1) para evitar erros por atraso (reserva de posicionamento para o controlador de posição)!

Sentido de rotação

O sentido de rotação é verificado através do eixo do motor. O rotor gira em sentido horário (para a direita) ou antihorário (para a esquerda). Os parâmetros P350 e P351 Change direction of rotation invertem esta definição.

P915 Speed feedforward

O parâmetro P915 somente é ativo quando o tipo de rampa "LINEAR" é ajustado (P916)! A função não tem efeito para os tipos de rampa SENOIDAL e QUADRÁTICA.

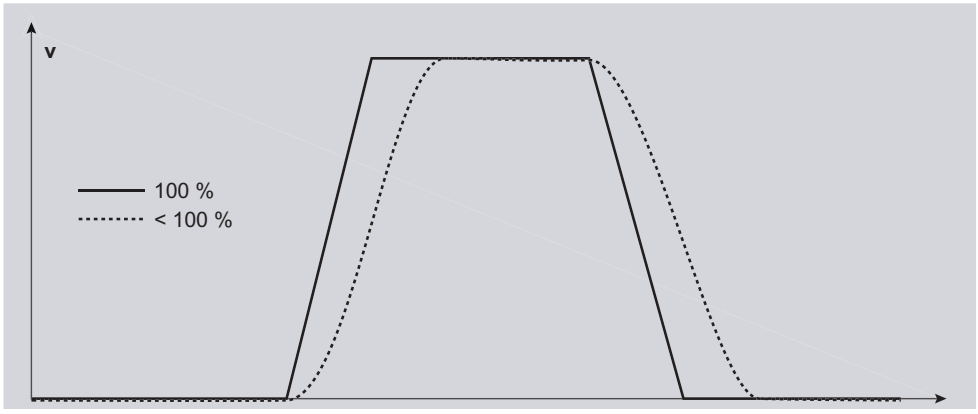
Somente é possível alcançar aceleração mais suave utilizando o parâmetro de pré-controle da rotação (P915) se a rotina do controle de rotação tiver um ajuste de baixa rigidez (ajuste para a colocação em operação <0.7) (ajuste com ganho P controlador de rotação (P200), constante de tempo do controle de rotação. (P201)). O P915 deve ser ajustado para um valor inferior a 100 %. Se for especificado um valor menor do que 100%, ocorre uma folga maior entre o ajuste de posição e a posição atual (distância de atraso) durante a operação de posicionamento.

Isto resulta em uma transição "mais suave" no início e no final da aceleração.

Tabela 3: Propriedades do pré-controle da rotação

Reação de controle desejada	Pré-controle da rotação	Vantagem	Desvantagem
Posicionamento com distância mínima de atraso	100 %	• Tempo mínimo de deslocamento • Menor distância de atraso	• Característica de torque em forma de bloco • Operação brusca para a posição final
Operação suave para a posição final	50 % ... 80 %	• Operação suave para a posição final	• Maior distância de atraso • Operação de posicionamento mais longa

O parâmetro P203 Filter accel. feedforward tem o mesmo efeito, mas não causa um aumento na distância de atraso porque o valor de controle para o ajuste de rotação já foi especificado pelo gerador de rampa do controlador de posição.



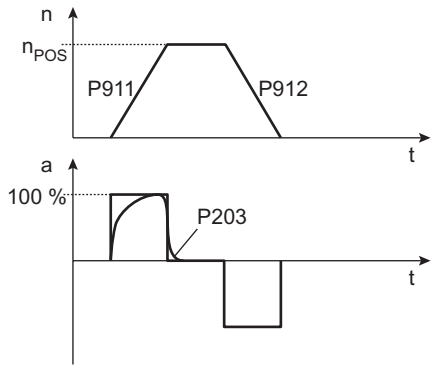
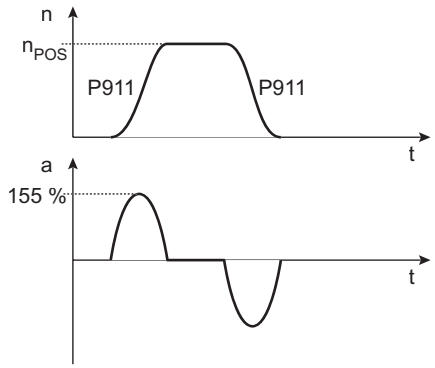
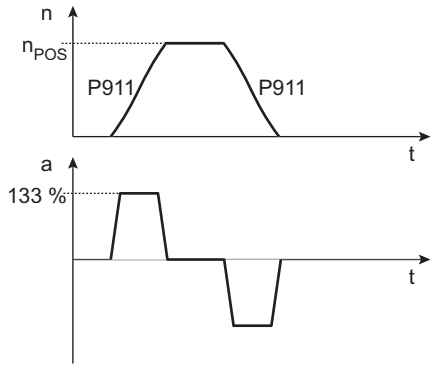
06016AXX

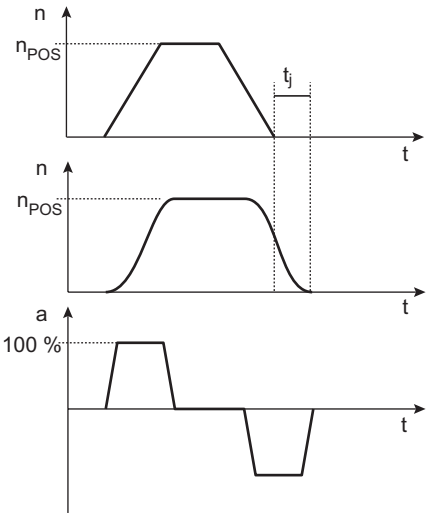
Figura 22: Efeitos do parâmetro P915 (pré-controle da rotação)

Faixa de ajuste: – 200 % ... **+ 100 %** ... + 200 %

Faz sentido utilizar os valores: + 50 % ... + 100 %

P916 Ramp type Este parâmetro especifica o tipo das rampas de posicionamento. Isto influencia a rotação e a aceleração durante o posicionamento. Se o sistema admite o tipo de rampa JERK LIMITED, este é o melhor tipo de rampa a utilizar em operações de posicionamento, utilizando o gerador de rampa interno.

Tipo de rampa	Características de posicionamento	Aplicações
LINEAR 	**Perfil da rotação** O perfil da rotação é linear e é especificado pelas rampas de posicionamento e pela rotação de posicionamento (trapezoidal). **Perfil da aceleração** O perfil da aceleração e, portanto, o perfil do torque é em forma de bloco (carga de solavanco). O parâmetro P203 Filter accel. feedforward reduz o movimento de solavanco, mas se o valor é muito alto, o conversor "move-se lentamente" para a posição final. O parâmetro **P915 Speed feedforward** pode ser utilizado para suavizar a rampa de posicionamento.	• Para acionamentos com qualidades dinâmicas baixas e pouca flexibilidade. • Operações de posicionamento rápido com rampas rápidas. Os materiais movidos são sujeitos à forte ação de solavanco (aumento súbito no torque). Este processo não é adequado para movimento de líquidos ou massas leves (por ex. em fábricas de engarrafamento).
SENOIDAL 	**Perfil da rotação** O perfil da rotação é similar ao seno^2 e é especificado pela rampa de posicionamento e rotação de posicionamento. **Perfil da aceleração** A aceleração é senoidal. A aceleração na partida e no final da rampa é menor comparada às rampas lineares. Utilizando o mesmo tempo de rampa como para aceleração linear, resulta em uma aceleração aprox. 1.55 maior na faixa média. Sem suavização de picos na partida do movimento.	• Operações de posicionamento de velocidade alta com acionamentos dinâmicos ou flexíveis. A aceleração e, portanto, o torque são 55% mais altos do que com a rampa linear. Durante a configuração, certifique-se que o motor e o conversor tenham o torque desejado (risco de erros por atraso).
QUADRÁTICA 	**Perfil da rotação** O perfil da rotação é quadrático no final da rampa e é especificado pela rampa de posicionamento e pela rotação de posicionamento. **Perfil da aceleração** A aceleração é trapezoidal. Isto significa que a ação de solavanco é menor do que para o tipo de rampa LINEAR. Este tipo de rampa é um meio-termo entre os requisistos de torque e o perfil de rampa suave. Sem suavização de picos na partida do movimento.	• Operações de posicionamento de velocidade alta com acionamentos dinâmicos ou flexíveis. A aceleração e, portanto, o torque são 33% mais altos do que com a rampa linear. Durante a configuração, certifique-se que o motor e o conversor tenham o torque desejado (risco de erros por atraso).
BUS RAMP	Um controlador externo escreve o ajuste de posição H499 SetpPosBus a cada 5 ms. Este valor é finamente intercalado em intervalos de 1 ms e especifica o valor nominal para o controle de posição.	Operação com controlador mestre.

Tipo de rampa	Características de posicionamento	Aplicações
JERK LIMITED 	A suavização de picos (P916) é baseada no princípio da rampa linear. Para limitação de choques mecâ-nicos, o torque e, portanto, a aceleração são trapezoidais para limitar a ação de solavanco. Os tempos que excederem, a suavização de picos acumula o torque na forma linear durante a aceleração, até o valor máximo ser al-cançado. Do mesmo modo, o torque é novamente reduzido em tempos extras na forma linear até zero. Isto significa que as vibrações do sistema podem ser virtualmente evitadas. Pode ser selecionada uma faixa de ajuste de 5 ms a 2000 ms (P933). O tempo de posicionamento em comparação à rampa linear é ampliado pelo ajuste do jerk time t_j. A aceleração e o torque não aumentam em comparação à rampa linear.	Todas as aplicações em que os tipos de rampa LINEAR, SENOIDAL ou QUADRÁTICA são realizados.
ELECTRONIC CAM	Ativa a versão aplicação "Came eletrônico".	
I SYNCHRONOUS OPERATION	Ativa a versão aplicação "Operação Síncrona Eletrônica."	

P917 Ramp mode Este parâmetro determina o uso do P912 Positioning ramp 2 quando o tipo de rampa está ajustado para LINEAR ou JERK LIMITED.

P917 = 0 (Modo 1)

A rampa de posicionamento 2 (P912) é utilizada somente para deslocamento direto até a posição final; todas as outras mudanças de velocidade são feitas com rampa de posicionamento 1 (P911).

P917 = 1 (Modo 2) = Separação da rampa de aceleração e de desaceleração

A aceleração é quando o conversor aumenta a velocidade, a desaceleração quando o conversor diminui. A rampa 1 é utilizada para aceleração, a rampa 2 para desaceleração.

A seqüência de movimento ocorre independente do ajuste no parâmetro P917 "RampModeLinear" durante uma operação simples de posicionamento (em parada), sem mudança dos dados de percurso durante o posicionamento.

No entanto, se um parâmetro de deslocamento (por exemplo, um novo final) é especificado durante o posicionamento, por exemplo, durante a rampa de frenagem que deve conduzir diretamente para o final, este conduz para uma diferença considerável, como é representado no diagrama abaixo.

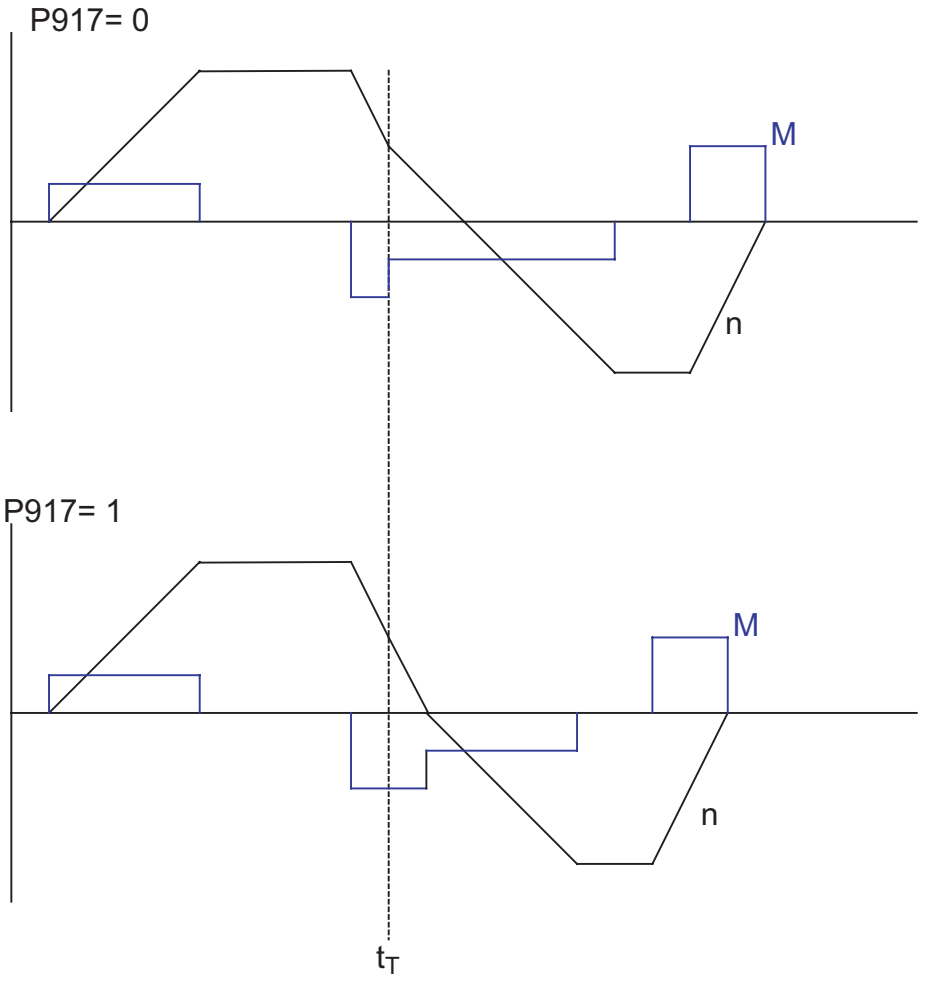


Figura 23: P917 Ramp mode

9.4 P92_ Monitoração IPOS

P920 / P921 Software limit switches

As chaves fim-de-curso do programa realizam uma função de segurança adicional definindo os limites da faixa de deslocamento. Os comandos de deslocamento não são realizados se suas posições finais (H492) estiverem fora das chaves fim-de-curso do software. Eles monitoram a posição do valor nominal e não o valor de posição atual. As chaves fim-de-curso do programa não tem efeito até o referenciamento ter sido realizado. Elas monitoram a posição do valor nominal, não o valor de posição atual.

Se o conversor já está em movimento, ele é desacelerado utilizando a rampa de parada de emergência. Nos dois casos, é gerada uma mensagem de irregularidade (F78, IPOS SW limit switches). O sistema responde executando uma parada de emergência e inibindo o estágio de saída. Deve ser realizado um reset.

Procedimento do reset:

- Sinal 1 em "reset input"
- Fonte de alimentação DESL / LIG (sem modo de backup 24 V)
- Reset manual via SHELL ("Parâmetro" / "Reset manual")
- Reset via palavra de controle IPOS$^{plus®}$ (variável do sistema H484)
- Reset via fieldbus

Depois do F78 (IPOS SW limit switches) ter sido reativado, a posição de referência não é mais definida. As chaves fim-de-curso do programa somente são ativadas quando o conversor é novamente referenciado! Se P941 Source actual position é ajustado para encoder absoluto DIP, ou se é utilizado um encoder do motor com Hiperface®, estes ajustes são imediatamente úteis sem referenciamento.

Desativação

Ajustar os parâmetros P920 e P921 em 0 (por ex. para operação de um sentido) para desativar as chaves fim-de-curso do programa.

P920 SW limit switch CW

Limita o deslocamento no sentido horário. O valor é dado nas unidades de deslocamento do usuário. A função monitoração da chave fim-de-curso do programa somente é ativada quando o conversor tiver sido referenciado. A monitoração é desativada se as chaves fim-de-curso antihorária e horária estiverem ajustadas em 0.

Faixa de ajuste: -2^{31} ... **0 unidades do usuário** ... $2^{31} - 1$ incrementos

P920 é idêntico ao H496.

P921 SW limit switch CCW

Limita o deslocamento no sentido antihorário. O valor é dado nas unidades de deslocamento do usuário. A função monitoração da chave fim-de-curso do programa somente é ativada quando o conversor tiver sido referenciado. A monitoração é desativada se as chaves fim-de-curso antihorária e horária estiverem ajustadas em 0.

Faixa de ajuste: -2^{31} ... **0 unidades do usuário** ... $2^{31} - 1$ incrementos

P921 é idêntico ao H497.

P922 Position window

A janela de posição define uma distância antes e depois da posição final (H492) de um comando de deslocamento ou parada (GOx ou ASTOP TARGET POSITION). Assim que a diferença entre o valor de posição atual e a posição final for menor do que a janela de posição, o sinal "IPOS IN POSITION" é gerado. Este sinal é disponível através de uma saída digital, que deve ser ajustada para a função "IPOS IN POSITION". Este valor também é disponível na variável do sistema H473 palavra de estado / bit 19. O sinal "IPOS IN POSITION" é imediatamente reativado no MOVIDRIVE® A quando um novo comando de deslocamento é enviado. Para MOVIDRIVE® B, o sinal "IPOS IN POSITION" é imediatamente reativado somente com o comando _GoAbs (), caso contrário ele é reativado após 1 ms.

A janela de posição sempre é monitorada desde que o modo de operação com IPOS esteja ativo (P700). A precisão de posicionamento não é afetada pelo valor da janela de posição.

Faixa de ajuste: 0 ... **50** ... 2^{15} – 1 incrementos

Mesmo se a função módulo está ativa, a unidade de incremento ainda refere-se ao encoder IPOS$^{plus®}$ (por exemplo, encoder do motor) não H455 ModActPos.

P922 é idêntico ao H493.

P923 Lag error window

A janela de erro por atraso define a diferença máxima permitida entre a posição do valor nominal atual, que o gerador de rampa especifica a cada 1 ms, e a posição atual. Se o valor especificado é excedido, a irregularidade F42 (lag error) é ativada. A reação à F42 pode ser ajustada utilizando o parâmetro P834 "Response LAG ERROR."

Desativação: Pode-se desativar a monitoração do erro por atraso pelo ajuste da janela P923 Lag error em 0.

Faixa de ajuste: 0 ... **5000** ... 2^{31} – 1 incrementos

Mesmo se a função módulo está ativa, a unidade de incremento ainda refere-se ao encoder IPOS$^{plus®}$ (por exemplo, encoder do motor) não H455 ModActPos.

P923 é idêntico ao H494.

9.5 P93x Funções especiais IPOS

P930 Override

Pode-se utilizar a função override para mudar a velocidade de deslocamento atual das operações de posicionamento na faixa de 0 a 150 % para cada velocidade programada. Para isto, é necessário uma entrada analógica, onde 0 a 150 % corresponde a 0...10 V. O valor máximo de velocidade é sempre limitado por P302 Maximum speed 1 / P312 Maximum speed 2 .

Faixa de ajuste: ON / OFF

P931 IPOS CTRL word Task 1

A palavra de controle IPOS Task 1 somente é disponível no controle manual DBG, não no SHELL.

Faixa de ajuste: START / STOP / HOLD

P931 pode ser ajustado para "HOLD" durante a operação; este pára **todos** os tasks. P931 pode ser ajustado para "STOP" durante a operação; este pára somente o task 1. O task 2 e o task 3 continuam.

P932 IPOS CTRL word Task 2

A palavra de controle IPOS Task 2 somente é disponível no controle manual DBG, não no SHELL.

Somente um parâmetro indicativo; não pode ser ajustado utilizando DBG.

Faixa do indicador: START / STOP

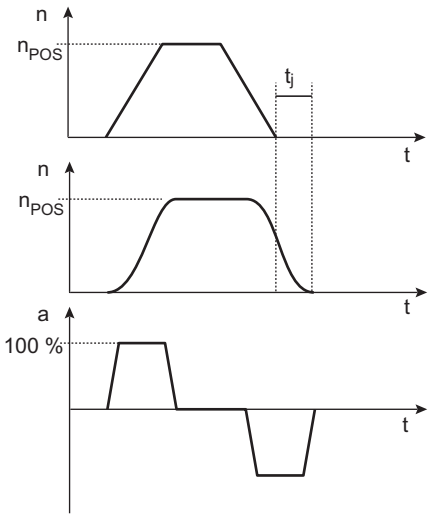
P932 é somente um parâmetro indicador utilizado para verificar se o task 2 está funcionando (partida / parada são os valores indicados).

P933Jerk time

O tempo de solavanco somente é útil quando P916 é ajustado para JERK LIMITED. Isto especifica o tempo que a aceleração ajustada com as Rampas de Posicionamento P911 e P912 tiver sido alcançado.

Se o tempo de solavanco é ajustado para um valor tão baixo que o valor limite da rampa de posicionamento é alcançado, o tempo de posicionamento é ampliado para um valor maior do que da rampa linear pelo tempo de solavanco ajustado (ver P916 Ramp type).

Faixa de ajuste: 0,005 ... 2 s



P938Speed task 1

Somente MOVIDRIVE® B

Faixa de ajuste: 0 ... 9 comandos/ms

O ajuste padrão para task 1 é 1 comando/ms. A velocidade pode ser aumentada até 9 comandos/ms com P938. P938 compartilha os recursos para o aumento da velocidade com P939.

Exemplo: P938 = 1: Task 1 é processado em 2 comandos/ms.

P939Speed task 2

Somente MOVIDRIVE® B

Faixa de ajuste: 0 ... 9 comandos/ms

O ajuste padrão para task 2 são 2 comandos/ms. A velocidade pode ser aumentada até 9 comandos/ms com P939. P939 compartilha os recursos para o aumento da velocidade com P938.

Exemplo: P939 = 1: Task 2 é processado em 3 comandos/ms.

9.6 P94x Encoder IPOS

P940 IPOS variables edit

As variáveis de edição IPOS somente são disponíveis com o controle manual DBG, não no SHELL.

Faixa de ajuste: ON / OFF

As variáveis IPOS$^{plus®}$ podem ser alteradas quando P940 = 'ON.'

P941 Source actual position

Faixa de ajuste: Encoder do motor (X15) / Encoder ext. (X14) / Encoder absoluto (DIP)

Define o encoder o qual o IPOS$^{plus®}$ posiciona.

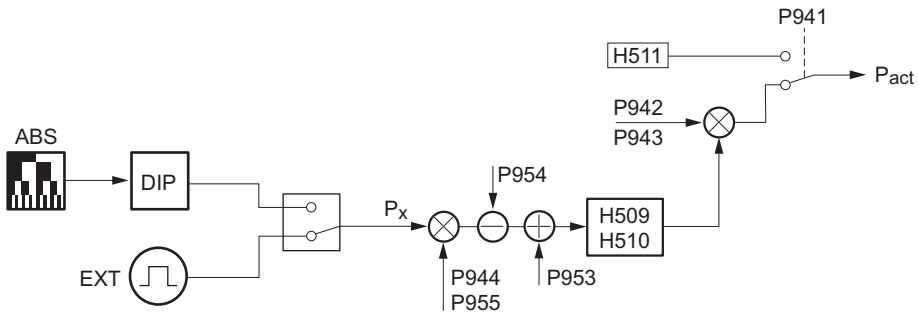
P942 / P943 Encoder factor numerator / denominator

Faixa de ajuste: 1 ... 32767

Com os parâmetros P942, P943 e P955 para encoder absoluto SSI DIP ou P944 para encoder externo, os incrementos do encoder do motor são comparados aos incrementos de posição do encoder externo para o gerador de rampa e o sistema de controle.

Praticamente, isto adapta a resolução física, a relação mecânica entre o encoder do motor e o encoder externo, e a taxa de alimentação mecânica constante (por exemplo, para encoders incrementais lineares externos).

O quociente é útil após H509 ou H510 ter sido criado.



P_x = Valor de posição não-linearizada do encoder
P_{act}= Valor de posição atual para gerador de rampa e controlador de posição
P941 = Fonte de posição atual
P942 = Fator numerador do encoder
P943 = Fator denominador do encoder
P944 = Escala do encoder ext.
P953 = Offset de posição
P954 = Offset zero
P955 = Escala do encoder DIP
H509 = Valor atual do encoder absoluto na variável
H510 = Valor atual do encoder externo na variável
H511 = Valor atual do encoder do motor na variável

Exemplo 1

Encoder do motor: 1024 pulsos /encoder externo: 512 pulsos

Relação mecânica entre o encoder do motor e o encoder externo: 1 : 1

1. Solução (recomendada): P942 = 1 / P 943 = 1 / P944 = x2

Valores em uma volta do motor:

H511 = 0, 1, 2 ... 4095, 4096

H510 = 0, 2, 4 ... 4094, 4096

2. Solução (não utilizar para rampas > 20 s e altas rotações do motor):

P 942 = 2 / P943 = 1 / P944 = x1

Valores em uma volta do motor:

H511 = 0, 1, 2 ... 4095, 4096

H510 = 0, 1, 2 ... 2047, 2048

Exemplo 2

Encoder do motor: 1024 pulsos /encoder externo: 1024 pulsos

Relação mecânica entre o encoder do motor e o encoder externo: 4 voltas : 3 voltas

1. Solução (recomendada): P942 = 4 / P943 = 3 / P944 = x1

Valores em uma volta do motor:

H511 = 0, 1, 2 ... 4095, 4096

H510 = 0, 1, 2 ... 3072

2. Solução (não utilizar para rampas > 20 s e altas rotações do motor):

P942 = 1 / P943 = 3 / P944 = x4

Valores em uma volta do motor:

H511 = 0, 1, 2 ... 4095, 4096

H510 = 0, 4, 8 ... 12288

Determinando os parâmetros

Procedimento para verificar ou determinar os parâmetros se a redução ou resolução for desconhecida:

- Determinar P944 ou P955.
- Anotar os valores das variáveis H509 ou H510 e H511 Posição atual do motor.
- Movimentar o conversor cerca de 30 000 incrementos (H511).
- Calcular a diferença entre os valores anotados e os novos valores das variáveis:
  - H509 antigo – H509 novo = Diferença H509 (H510 antigo – H510 novo = Diferença H510)
  - H511 antigo – H511 novo = Diferença H511
- Os valores não devem diferenciar-se por mais do que 32 767 (2^{15} –1). Se os valores forem maiores, dividir os diferenciais pelo mesmo número para obter valores correspondentemente menores. Por outro lado, repetir o procedimento com uma distância de percurso mais curta.
- Inserir o resultado da diferença H511 no P942 Fator nominador do encoder e H509 no P943 Fator denominador do encoder.

O ajuste tem as seguintes funções:

- Realiza uma verificação entre os dois valores de posição (mensagem de irregularidade F95, irregularidade considerável).
- Adapta corretamente as rampas de posicionamento e a velocidade de posicionamento para o segundo encoder.
- Avalia corretamente os valores para a colocação em operação, que incluem, por ex.:
  - Pré-controle da rotação
  - Pré-controle do torque
  - Filtro
  - Componente P

Um ajuste impreciso conduz a uma discrepância nas rampas de posicionamento e na velocidade de posicionamento. Isto também pode causar as mensagens de irregularidade P95 Plausibility error ou F08 n-monitoring.

P942 e P943 são calculados durante a colocação em operação SHELL na DIP.

P944 Encoder scaling ext. encoder

O parâmetro aumenta o valor de posição não-linearizado do encoder externo, antes de ser reproduzido na variável H510.

Faixa de ajuste: ×1 / ×2 / ×4 / ×8 / ×16 / ×32 / ×64

É adaptado a importância da resolução de deslocamento do encoder do motor e do encoder externo. O parâmetro é ajustado assim que a relação da informação de deslocamento entre o encoder do motor e o encoder externo estiver mais próxima possível de "1". Ajustar o parâmetro inicialmente para "×1". Para isto, observar os valores nas variáveis H510 e H511.

Movimentar o conversor cerca de 1000 incrementos (H511). Determinar a diferença entre o valor anotado e o valor atual e calcular o quociente. Ajustar o parâmetro P944 Encoder scaling ext. encoder para o valor mais próximo do quociente calculado.

Importante: A escala do encoder influencia diretamente os parâmetros P900 Reference offset, P942 Encoder factor numerator e P943 Encoder factor denominator e o grupo de parâmetro P92x IPOS monitoring. Todas as posições do programa IPOS$^{plus®}$ tem que ser ajustadas quando utilizar o encoder externo. Deve ser feito o ajuste de todos os parâmetros indicados, toda vez que a escala do encoder for mudada.

Obs: Se o fator x2 é ajustado, por exemplo, quando P941 = EXT. ENCODER, o IPOS somente pode ser utilizado para a posição 0, 2, 4, 6 ... incrementos (ver também P942 / P943).

P945 Synchronous encoder type (X14)

Inserir o tipo do encoder externo. Os possíveis tipos de encoder são (dependendo do conversor):

- TTL: Encoder com sinal de saída digital, retangular (Nível TTL 0 V, 5 V)
- SIN/COS: Encoder com sinal de saída analógico, em forma senoidal (1 V_{SS})
- HTL: Encoder com sinal de saída digital, retangular (Nível HTL 0 V, 24 V)
- HIPERFACE: Encoder com interface Hiperface® / encoders absolutos de uma volta ou de múltiplas voltas

Tipo do encoder SEW	Tipo de encoder dos parâmetros para a colocação em operação / contagem PPR do encoder
ES1S / ES2S / EV1S	SINE ENCODER / 1024
ES1R / ES2R / EV1R	INCREM. TTL ENCODER TTL / 1024
ES1T[1] / ES2T[1] / EV1T[1]	INCREM. TTL ENCODER TTL / 1024
ES1C / ES2C / EV1C	INCREM. TTL ENCODER HTL / 1024
AV1H / AS1H / ES1H / EV1H	HIPERFACE

1) somente via DWI11A

P946 Synchronous encoder counting direction (X14)

Faixa de ajuste

: NORMAL / INVERTED

Define o sentido de contagem do encoder síncrono. O ajuste deve ser feito assim que combinar o sentido de contagem do encoder do motor (X15) e do encoder síncrono (X14).

P947Hiperface® offset X14

Faixa de ajuste: –(2^{31}–1) ... 0 ... 2^{31}–1

Este parâmetro é utilizado para determinar o ponto zero do encoder externo para um encoder Hiperface®.

Utilizar este parâmetro para definir o zero da máquina sem referenciamento. Isto adiciona ou subtrai o offset do valor do encoder.

- P947 Hiperface® offset X14 tem efeito na posição atual do encoder externo H510.

  H510 = Valor do encoder – P947

A posição atual é determinada diretamente após os valores terem sido inseridos. Isto não necessita referenciamento anterior.

Observe:

No caso do referenciamento para um sistema de conversor com encoder Hiperface®, o offset Hiperface® P947 é recalculado e regravado pelo referenciamento quando P941 Source actual position = EXT. ENCODER.

Aplica-se o seguinte, quando P960 Modulo function = OFF:

- P947 = Valor do encoder – P900

Quando a função módulo está ativa, P900 refere-se aos incrementos de carga ajustados (ver "P96x IPOS Modulo function").

9.7 P95x DIP

Os parâmetros DIP são descritos detalhadamente no manual "Positioning with Absolute Encoder and Absolute Encoder Interface DIP11A". A DIP não pode ser utilizada com BG0.

P950 Encoder type

Selecionar o encoder absoluto SSI conectado ao X62 da DIP11. A partir do firmware e da DIP versão .15, são realizados pelo MOVIDRIVE® A os seguintes sistemas de encoder:

- NO ENCODER / Função desligada
- VISOLUX EDM / Instrumento de medição de distância a laser
- T&R CE65, CE100 MSSI / Encoder rotativo
- T&R LE100 / Instrumento de medição de distância a laser
- T&R LA66K / Sensor de distância linear
- AV1Y / ROQ424 / Encoder rotativo
- STEGMANN AG100 MSSI / Encoder rotativo
- SICK DME-3000-111 / Instrumento de medição de distância a laser
- STAHLWCS2-LS311 / Escala linear metálica
- SICK ATM60 / ATM90 / Encoder rotativo
- STEGMANN AG 626 / Encoder rotativo
- IVO GM401 / Encoder rotativo
- STAHL WCS3 / Escala linear metálica
- LEUZE OMS1 / Instrumento de medição de distância a laser
- T&R ZE 65M / Encoder rotativo
- LEUZE BPS37 / Escala linear por código de barra

P951 Counting direction

Faixa de ajuste: NORMAL / INVERTED

Define o sentido de contagem do encoder absoluto. O ajuste deve ser feito assim que combinar o sentido de contagem do encoder do motor (X15) e do encoder absoluto (X62).

P952 Cycle frequency

Faixa de ajuste: 1 ... 200 %

Define a freqüência de chaveamento a qual a informação do encoder absoluto é transmitida do encoder para o conversor. A freqüência de chaveamento de 100 % corresponde à freqüência nominal do encoder em relação a um cabo de 100 m de comprimento.

P953 Position offset

Faixa de ajuste: $-(2^{31}-1)$ ... 0 ... $2^{31}-1$

Este parâmetro adapta a faixa de registro do encoder para a faixa de trabalho da máquina.

O offset de posição P953 precisa ser ajustado somente para encoders rotativos; para outros encoders, deve estar ajustado para 0.

Este parâmetro é utilizado para a mudança de posição do encoder absoluto para uma posição fora da faixa de trabalho. Movimentar o conversor aproximadamente para o centro da faixa de trabalho.

Se a mensagem de irregularidade F92 DIP Registration range aparecer quando isto estiver sendo feito, ignore-a com um reset e continue com a colocação em operação. Ler o valor da variável H509 ACT.POS.ABS e inserir o seguinte valor no parâmetro P953 Position offset: P953 = Variável H509 – 0. 5 x faixa de registro.

Observe: O valor de posição será recalculado e regravado automaticamente após o referenciamento ter sido completado com sucesso.

P954 Zero offset

Faixa de ajuste: $-(2^{31}-1)$ ... 0 ... $2^{31}-1$

O offset zero é utilizado para fixar o valor desejado para uma posição específica. A faixa de valores pode adotar valores de posição positivo ou negativo. O parâmetro máximo válido não deve ser excedido. O limite é determinado pela faixa de valores do numerador (2^{31}) e pela faixa de valores do encoder absoluto. Movimentar o conversor para uma posição conhecida. Ler o valor da variável H509 ACT.POS.ABS e inserir o seguinte valor no parâmetro P954 Zero offset: P954 = Variável H509 – valor desejado.

O valor desejado é o valor indicado que se deseja ter para a posição atual.

P955 Encoder scaling

Faixa de ajuste: ×1 / ×2 / ×4 / ×8 / ×16 / ×32 / ×64

O parâmetro aumenta o valor de posição do encoder externo DIP antes de ser reproduzido na variável H509. Este parâmetro também afeta P953 Position offset e P954 Zero offset.

O parâmetro é ajustado de modo que a relação da informação de deslocamento entre o encoder do motor e o encoder absoluto seja mais próxima possível de "1".

Se o encoder do motor não estiver conectado, este parâmetro é irrelevante.

A escala do encoder influencia diretamente os parâmetros P953 Position offset, P954 Zero point offset, P942 Encoder factor numerator, P943 Encoder factor denominator e o grupo de parâmetro P92x IPOS monitoring. Deve ser feito o ajuste de todos os parâmetros indicados, toda vez que a escala do encoder for mudada.

9.8 P96x Função Módulo IPOS$^{plus®}$

A função módulo IPOS$^{plus®}$ é utilizada para posicionamento contínuo, por exemplo com posicionamento de mesas giratórias ou transportadores de corrente. Para informação mais detalhada, consultar o manual IPOS$^{plus®}$, seção Detecção da Posição e Posicionamento / Função Módulo.

P960 Modulo function

- OFF: A função módulo está desligada.
- SHORT: A função módulo "short travel" está ativa. O conversor movimenta-se da posição atual para a posição final pelo caminho mais curto possível. São possíveis ambos sentidos de rotação.
- CW: A função módulo "Horária" está ativa. O conversor movimenta-se da sua posição atual para a posição final com um sentido de rotação "Horário", mesmo que isso signifique movimentar por uma distância maior. Não é possível o sentido de rotação "Antihorário".
- CCW: A função módulo "Antihorária" está ativa. O conversor movimenta-se da sua posição atual para a posição final com um sentido de rotação "Antihorário", mesmo que isso signifique movimentar por uma distância maior. Não é possível o sentido de rotação "Horário".

Relação de redução do redutor i

A redução dos redutores SEW normalmente é arredondada para 2 casas decimais na placa de identificação. Se este é o caso, consultar a SEW para descobrir a redução exata do número de dentes, ou ler o valor da placa de identificação eletrônica do encoder (somente com Hiperface®).

Exemplo: K37CM71S ... com i_{tot} = 24.99 tem uma relação de transmissão de 9021 / 361. P961 = 9021 / P962 = 361.

P961 Modulo numerator

Faixa de ajuste: 0 ... 2^{31}

Simulação do redutor inserindo o número de dentes do redutor e o redutor adicional.

Módulo numerador = Numerador do redutor i × numerador do redutor adicional i

P962 Modulo denominator

Faixa de ajuste: 0 ... 2^{31}

Simulação do redutor inserindo o número de dentes do redutor e o redutor adicional.

Módulo denominador = Denominador do redutor i × denominador do redutor adicional i

P963 Modulo encoder resolution

Faixa de ajuste: 0 ... 4096 ... 20 000

Resolução do sistema de encoder IPOS$^{plus®}$ selecionado em incrementos.

A resolução do encoder IPOS$^{plus®}$ para posicionamento do encoder do motor é ajustada para 4096 incrementos (o pré-requisito é uma volta do encoder de 512 a 2048).

Para utilizar a função módulo, P963 × P961 deve ser < 2^{31} (ver também a seção Função Módulo).

10 Compiler – Editor

10.1 Características técnicas

- Controle seqüencial e de posicionamento integrado IPOS$^{plus®}$. As unidades IPOS$^{plus®}$ não necessitam qualquer hardware adicional.
- Criação do programa em uma linguagem de alto nível
- Nomes simbólicos das variáveis
- Possibilidade de criar módulos de programa que podem ser utilizados em outros projetos
- Programação clara, modular e estruturada
- Várias técnicas de rotina
- Controle compiler utilizando comandos pré-processadores
- Estruturas padrão
- Estruturas definidas pelo usuário
- Acesso a todas as variáveis do sistema
- Funções padrão
- Debugger para solução de problemas
- Amplos opcionais para comentários
- Integrado na interface Windows
- Integrado no pacote do programa MOVITOOLS®

10.2 Primeira fase

Esta descrição pretende ajudá-lo a conhecer a programação do compilador IPOS$^{plus®}$ o mais rápido possível. Será dado uma introdução nas funções básicas do compilador, por meio de um exemplo que é criado e desenvolvido passo-a-passo de um capítulo ao outro.

Esta introdução é dividida em três fases com os tópicos mais importantes.

Fase 1: Criando um projeto

Este capítulo mostra como iniciar o compilador IPOS$^{plus®}$ e como criar um novo projeto.

Fase 2: O primeiro programa IPOS$^{plus®}$

Durante este capítulo, você criará o seu primeiro programa IPOS$^{plus®}$.

Fase 3: Compilando e iniciando o programa IPOS$^{plus®}$

Neste capítulo, você irá compilar o programa criado na fase 2, carregá-lo no MOVIDRIVE® e operá-lo.

Pré-requisitos

Esta introdução presume que você conhece o sistema operacional Windows 95, Windows 98, Windows NT ou Windows 2000 e a operação geral dos programas Windows.

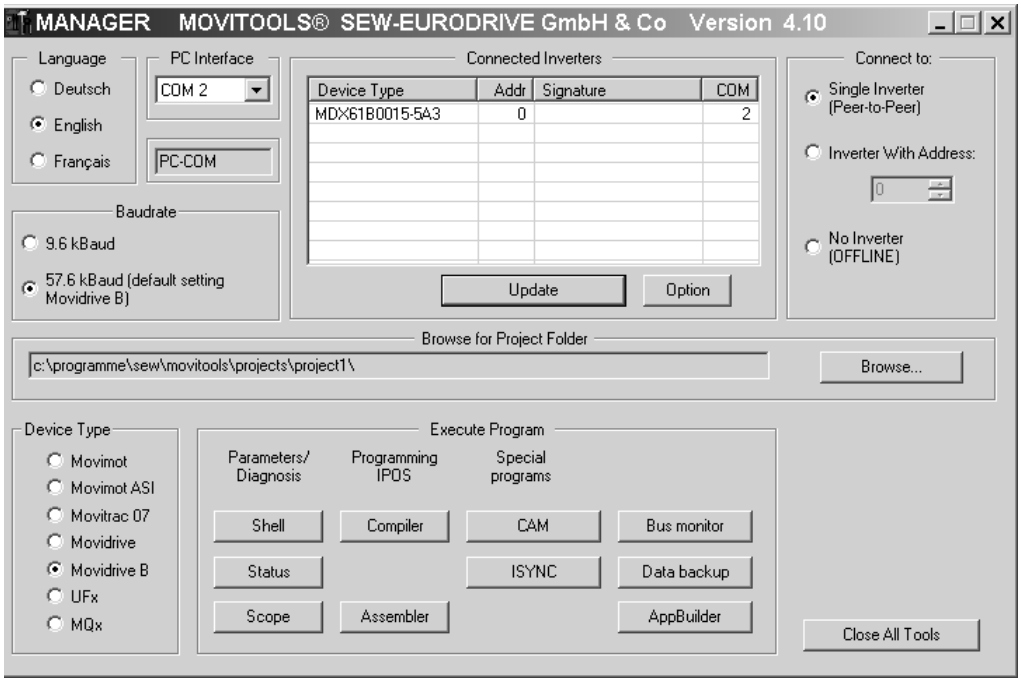
Você também deve ter o MOVITOOLS® 4.10 ou mais novo instalado.

Fase 1: Criando um projeto

Este capítulo mostra como iniciar o compilador IPOS$^{plus®}$ e como criar um novo projeto.

Iniciando o compilador IPOS$^{plus®}$

O compilador IPOS$^{plus®}$ é iniciado a partir do MOVITOOLS® Manager. O MOVITOOLS® Manager está localizado no menu para colocação em operação do Windows em Programs / MOVITOOLS x.xx / MT Manager.



10453AEN

Figura 24: MOVITOOLS® Manager

Conectar o MOVIDRIVE® A via USS11A e o MOVIDRIVE® B via UWS21A através de uma porta serial livre no seu PC, utilizando um cabo serial nulo do modem. Selecionar esta porta no PC COM Interface group. Na figura acima, COM 2 é a porta selecionada.

A taxa de transmissão é ajustada de fábrica para 9600 baud. Se estiver trabalhando com um MOVIDRIVE® B, você pode mudar a taxa de transmissão para 57 kBaud no campo "Baud rate". A taxa de transmissão ajustada no MOVITOOLS® Manager deve combinar o ajuste no MOVIDRIVE® (Chave DIP S13).

O MOVIDRIVE® deve ser conectado à rede ou à uma fonte de alimentação externa de 24 V para permitir a comunicação da unidade via porta serial.

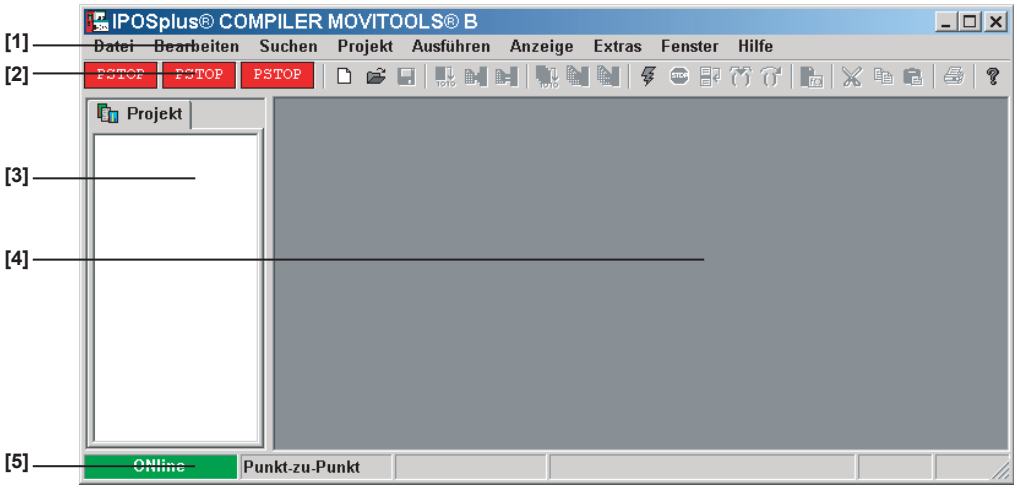
Clicar na tecla [Update] para iniciar a busca de todas as unidades conectadas. As unidades encontradas pelo sistema são indicadas na lista acima da tecla. A sua unidade deve estar indicada na lista. Se não estiver, pode não haver conexão entre o PC e o conversor. Verificar esta conexão.

Selecionar uma unidade da lista para ajustar o endereço adequado e comutar para o modo online.

Iniciar o compilador IPOS$^{plus®}$ clicando na tecla [Compiler].

Descrição geral da interface IPOS$^{plus®}$

A interface do programa a seguir é indicada quando iniciar o compilador IPOS$^{plus®}$.



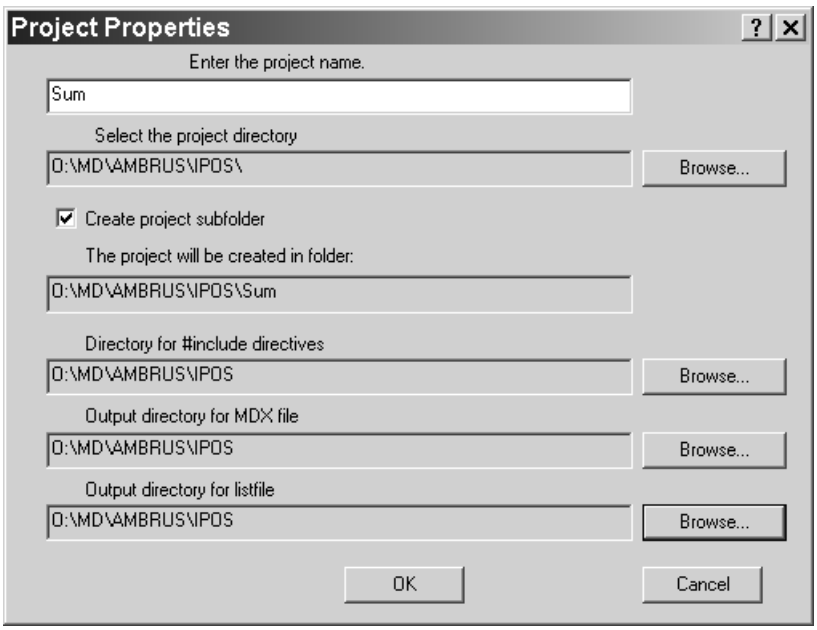
53555AXX

Figura 25: Áreas da interface de programação

[1] Barra do menu
[2] Barra de ferramentas
[3] Janela do projeto
[4] Janela principal
[5] Barra de estado

Criando um novo projeto

Para criar um novo projeto, escolher [Project] / [Create new...] da barra do menu. Aparece uma caixa de diálogo a qual você deve especificar as propriedades gerais do projeto.



10455AEN

Figura 26: Criando um novo projeto

A primeira linha contém o nome do projeto. Dar ao seu projeto um nome original que você reconhecerá no futuro.

A segunda linha especifica o diretório o qual o projeto deve ser salvo. Escolher o diretório utilizando a tecla [Browse]. O diretório já deve existir.

Se deve ser criada uma subpasta com o nome do projeto no caminho especificado, você deve marcar a caixa "Create project subfolder". O arquivo do projeto é então armazenado na subpasta.

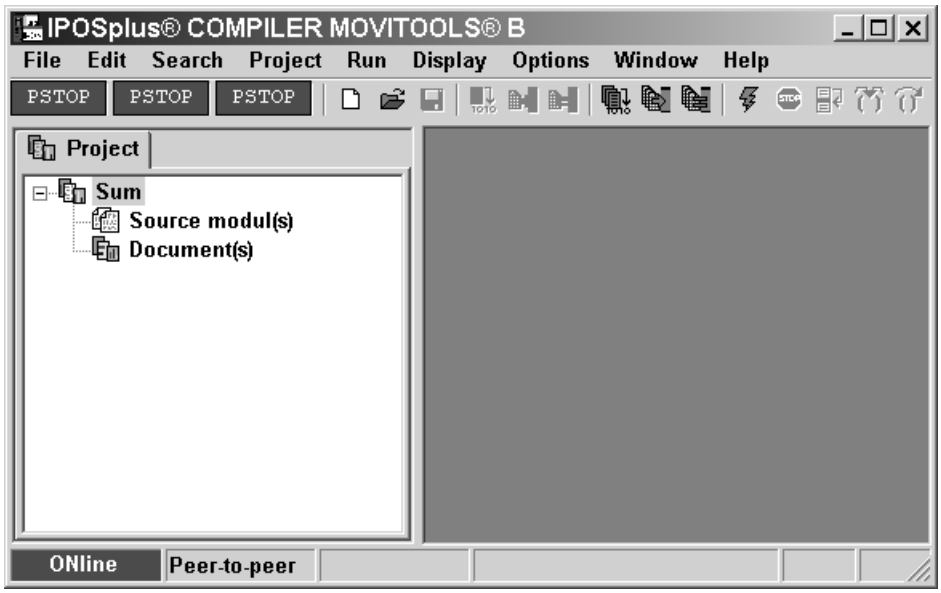
A quarta linha especifica o diretório o qual o compilador procura pelos arquivos. Estes arquivos são adicionados aos arquivos de texto fonte utilizando uma indicação #inclusa. O diretório criado durante a instalação é inserido aqui como padrão, por exemplo: c:\programme\sew\movitools\projects\include.

As linhas 5 e 6 especificam os diretórios os quais o arquivo MDX (arquivo com o programa IPOS$^{plus®}$) e o arquivo da lista (arquivo com informação adicional do programa) são criados. Estes arquivos somente são criados se você tiver verificado as caixas adequadas em [Extras] / [Settings] / [Compiler].

Depois de ter confirmado suas entradas clicando [OK], o compilador realiza os seguintes passos:

- Cria a pasta Total no diretório especificado (somente se você selecionou a opção "Create project subfolder").
- Cria um arquivo do projeto com o nome Total.icp na pasta Total.
- Fecha a caixa de diálogo.

O projeto agora aparece como uma árvore hierárquica na janela de projeto do programa.



10456AEN

Figura 27: Meu primeiro projeto

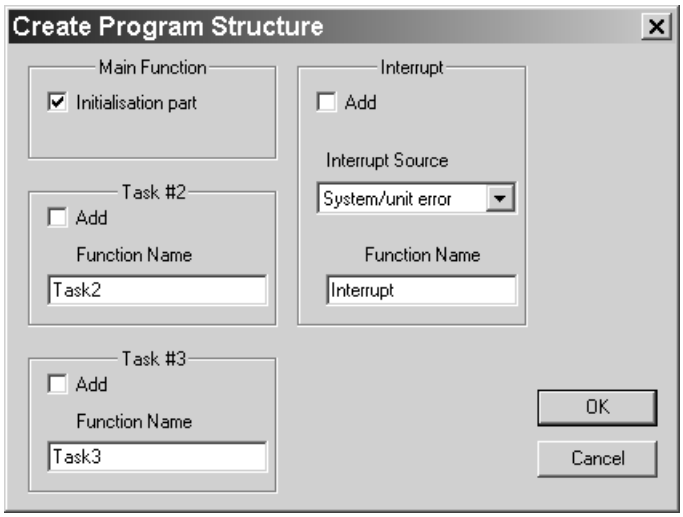
O próximo passo é criar um novo arquivo de texto fonte e acrescentá-lo ao projeto. Para isto, escolher [File] / [New] / [Source file..].

Aparece a seguinte caixa de diálogo:

10457AEN

Figura 28: Caixa de diálogo "File new"

Confirmar esta caixa de diálogo com [Yes]. Um novo arquivo fonte é acrescentado ao projeto. Agora é pedido a você entrar com um nome para o novo arquivo fonte. Entrar com o nome `summe.ipc`. Quando você clicar [Save] aparece outra janela. Você cria a estrutura do programa aqui.



10458AEN

Figura 29: Definindo a estrutura do programa

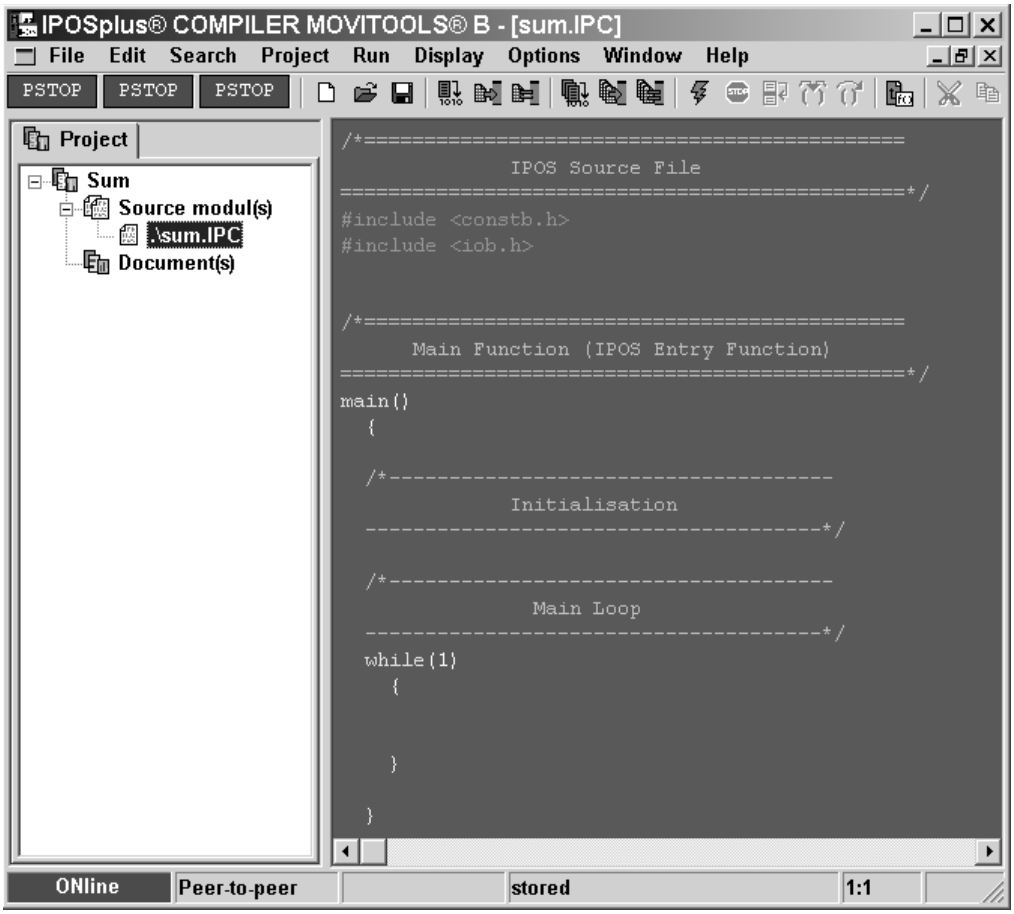
Na caixa de grupo [Main function], selecionar a caixa de verificação [Initialisation part]. Depois de ter deixado a caixa de diálogo confirmando sua entrada com [OK], a função "Main" é gerada automaticamente com a parte da inicialização. O arquivo fonte agora tem o seguinte conteúdo:

```
/*============================================
IPOS source file
============================================*/
#include <constb.h>
#include <iob.h>

/*============================================
Main function (IPOS initial function)
============================================*/
main()
{

  /*-------------------------------------
  Initialization
  -------------------------------------*/

  /*-------------------------------------
  Main program loop
  -------------------------------------*/
  while(1)
  {

  }

}
```

O comando `#include <const.h>` para MOVIDRIVE® A e `#include <constb.h>` para MOVIDRIVE® B insere o arquivo de cabeçalho, o qual são definidos os argumentos de todas as funções do sistema. O comando `#include <io.h>` para MOVIDRIVE® A e `#include <iob.h>` para MOVIDRIVE® B insere um arquivo que contém as definições das entradas e saídas digitais. Estas constantes e definições podem ser acessadas diretamente durante a programação.

A função "Main" contém uma parte da inicialização e a rotina principal do programa. Este é um programa correto que podia funcionar, mas não funciona, no entanto, pode conter quaisquer funções.

A janela do programa agora aparece assim:



10459AEN

Figura 30: Janela com um programa

Se fizer mudanças no texto fonte, salvar o projeto utilizando [File] / [Save All]. Fechar o programa escolhendo [File] / [Exit].

Fase 2: O primeiro programa IPOS$^{plus®}$

Neste capítulo, você criará seu primeiro programa IPOS$^{plus®}$.

Editando o programa IPOS$^{plus®}$

Iniciar novamente o compilador. Esta vez, o projeto e o arquivo `summe.ipc` são carregados automaticamente como foram abertos quando você saiu do programa.

Para saber todas as funções adicionais do compilador IPOS$^{plus®}$, você escreverá agora um programa que adiciona todos os números de 1 a 100 juntos.

Melhor do que utilizar a fórmula (n+1)×(n / 2) para isto, você deve programar uma rotina que soma o total por iteração.

O programa deve, portanto, ter a seguinte estrutura:

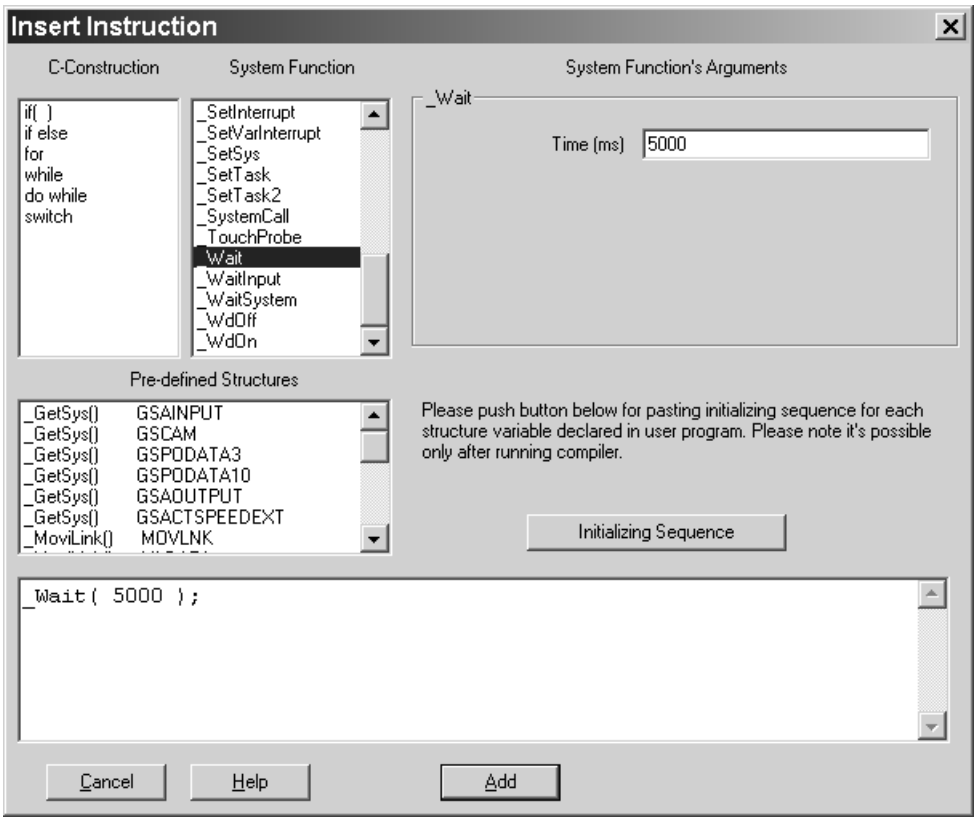
10460AEN

Figura 31: Programa de adição

O estado "Changed" é indicado na barra de estado. Isto significa que o programa foi mudado comparado com seu estado quando foi salvo pela última vez. Salvar o programa. O estado "Saved" é agora indicado na barra de estado.

É utilizado um destaque para que as palavras com diferentes significados sejam mostradas em cores diferentes oferecendo melhor visualização. Por exemplo, todos os nomes que o compilador reconhece (palavras-chave) são mostrados em amarelo. As funções do sistema fornecidas pelo MOVIDRIVE® são destacadas em azul.

Pode-se utilizar a ferramenta de inserção enquanto edita o programa. Clicar com o botão direito do mouse para abrir o conteúdo do menu que contém o item [Insert Instruction...]. Este item do menu permite acessar a ferramenta de inserção.



10461AEN

Figura 32: Ferramenta de inserção

Pode-se utilizar a ferramenta de inserção para selecionar várias construções C, funções do sistema e estruturas pré-definidas. Quando selecionar uma função do sistema, inserir os argumentos desta função na caixa de grupo à direita da janela. Utilizar [Add] para inserir o comando relevante para a posição onde o cursor foi colocado no texto fonte.

Para inserir a função _wait, selecionar primeiro a função _wait na lista das funções do sistema. O lado direito da janela indica os argumentos relativos a função correspondente. Para nosso exemplo, inserir o valor 5000 (representa 5000 ms).

Para informação adicional sobre a construção C ou a função do sistema, selecionar simplesmente o termo em questão em uma das duas listas e pressionar a tecla <F1 key> ou [Help].

Pode-se ativar a função ajuda a partir do texto fonte, colocando o cursor na palavra-chave _wait e pressionando a tecla <F1>.

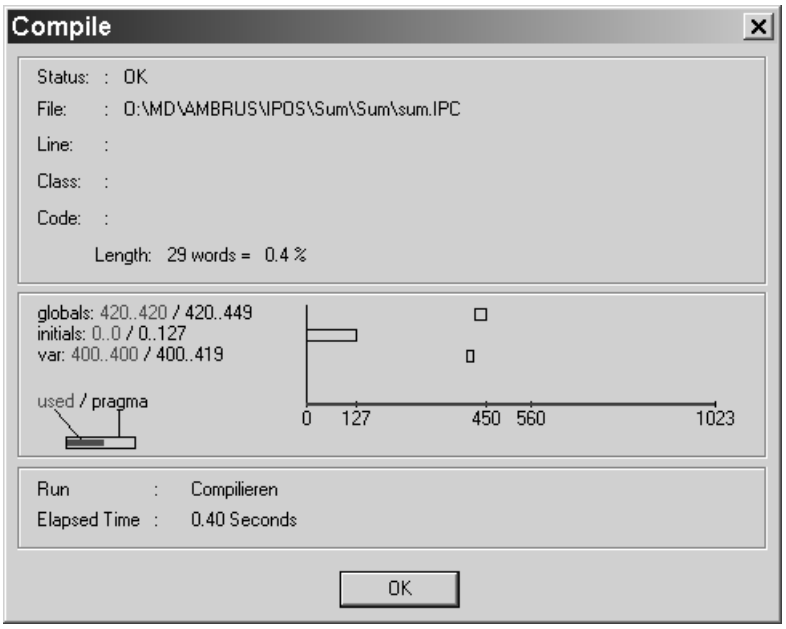
O compilador é suscetível, o que quer dizer que pode ter 2 variáveis diferentes MYVAR e myvar. Não são permitidas palavras acentuadas.

Fase 3: Compilando e iniciando o programa

Neste capítulo, você irá compilar o programa criado na fase 2, carregá-lo no MOVIDRIVE® e operá-lo.

Compilando o programa

Para gerar um programa de forma que o conversor possa entender, o projeto deve ser compilado. Para isto, pressionar o ícone ou escolher [Project] / [Compile].



10462AEN

Figura 33: Janela de mensagem indicada após a compilação

A janela de mensagem indicada acima aparece após o projeto ter sido compilado. Se o programa não tiver erros, é atribuído o estado OK. O tamanho do programa também é importante. Ele é especificado, como o comprimento das palavras-chave utilizado no código assembler. Este número absoluto também é convertido a uma porcentagem que especifica quanto espaço de memória é utilizado no IPOS$^{plus®}$.

O processo de compilação foi bem sucedido para o nosso programa. O programa tem 29 palavras IPOS no tamanho; isto é, ele ocupa 0.4 porcento da capacidade inteira de memória IPOS$^{plus®}$.

Fechar a janela escolhendo [OK].

Mensagens de erro durante a compilação

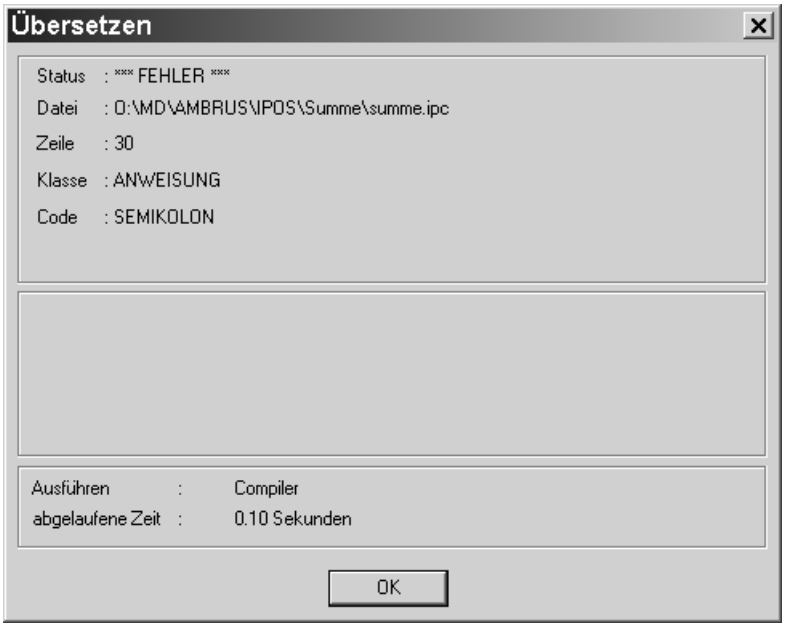
Como os erros de sintaxe podem ocorrer durante a programação, foi integrado um sistema de relatório do erro no compilador IPOS$^{plus®}$. Se o programa identifica um erro, ele indica a linha e gera uma mensagem de irregularidade correspondente classificando o erro.

Vamos tentar com um exemplo.

Mudar a rotina FOR como segue:

```
while(1)
{
  for (H1 = 0; H1 <= 100 )
  {
    // Create total
    H2 = H2 + H1;
    // Wait 5 s
    _Wait( 5000 );
  }
}
```

Neste exemplo, está faltando o terceiro argumento da rotina FOR. Quando você compila o sistema utilizando o ícone , aparece a seguinte mensagem:



10307AXX

Figura 34: Mensagem de erro durante a compilação

O estado ERROR é liberado como se um erro tivesse ocorrido. A linha, a classificação do erro e o código do erro também são indicados. Além disso, a linha do programa no arquivo que contém o erro é destacada com uma barra vermelha.

Clicar na tecla [OK] e corrigir o erro. O processo de compilação deve ser novamente repetido, depois de ter corrigido o erro.

Retificar o erro corrigindo a rotina FOR como segue:

```
while(1)
{
  for (H1 = 0; H1 <= 100;H1++)
  {
    // Create total
    H2 = H2 + H1;
    // Wait 5 s
    _Wait( 5000 );
  }
}
```

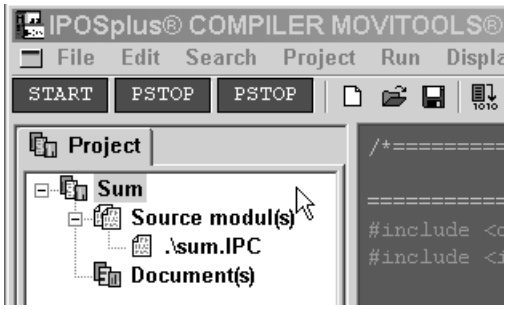
Carregando o programa no MOVIDRIVE®

Na próxima fase, o programa deve ser carregado no conversor. Para isto, pressionar o ícone ou escolher [Compile and download]. O programa é novamente compilado e carregado no conversor, depois de ter sido compilado com sucesso.

A primeira linha do programa na função "Main" é marcada com uma barra azul clara, depois do programa ter sido "baixado" com sucesso. Agora você pode iniciar o programa.

Iniciando e parando o programa

Pode-se iniciar o programa clicando no ícone . O programa funciona no MOVIDRIVE® e o estado START é indicado na barra de ferramentas.



10464AEN

Figura 35: Estado START

Ao mesmo tempo, a barra azul clara no programa é apagada.

Você pode ver que o programa está sendo processado. Neste pequeno programa de teste, a variável H2 é incrementada em fases de 5 s.

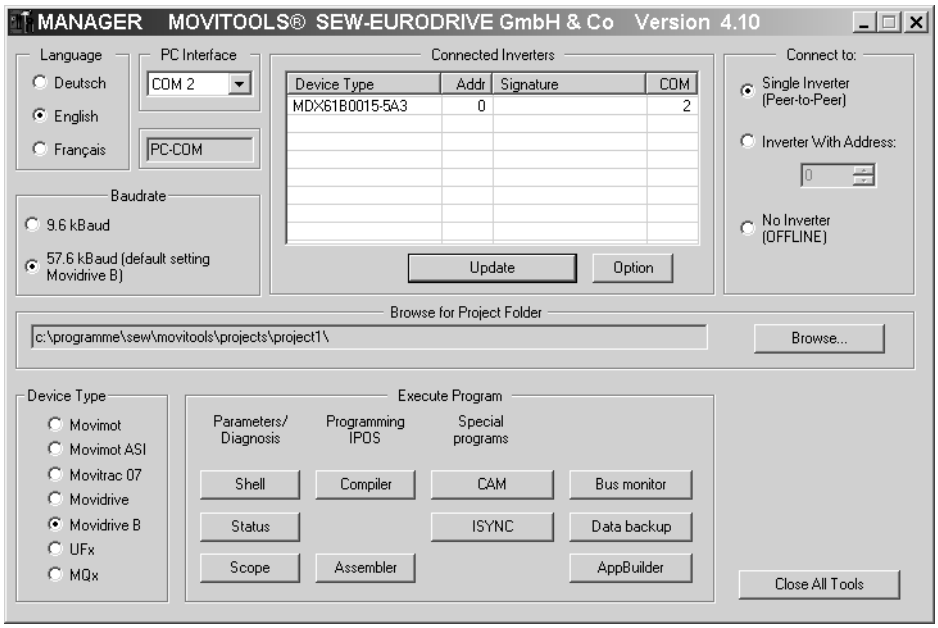
Para ver isto, abrir a janela da variável escolhendo [Display] / [All variables]. Agora, você pode observar a variável H2.

Agora, nós queremos parar o programa. Isto é feito pressionando o ícone . Depois disso, a primeira linha do programa na função "Main" é marcada com uma barra azul clara.

10.3 Compiler – editor

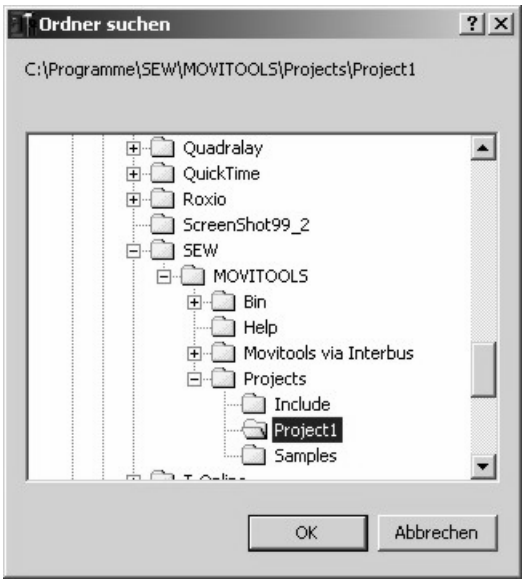
O compilador IPOS$^{plus®}$ é parte do pacote do programa MOVITOOLS® que foi desenvolvido como uma interface do usuário para a linha MOVIDRIVE®. O compilador IPOS$^{plus®}$ é iniciado a partir do MOVITOOLS® Manager.

Se você escolher um diretório de trabalho a partir do gerenciamento do projeto antes de iniciar o compilador IPOS$^{plus®}$, este diretório de trabalho é utilizado como o caminho direto quando você carrega e salva dados.



10453AEN

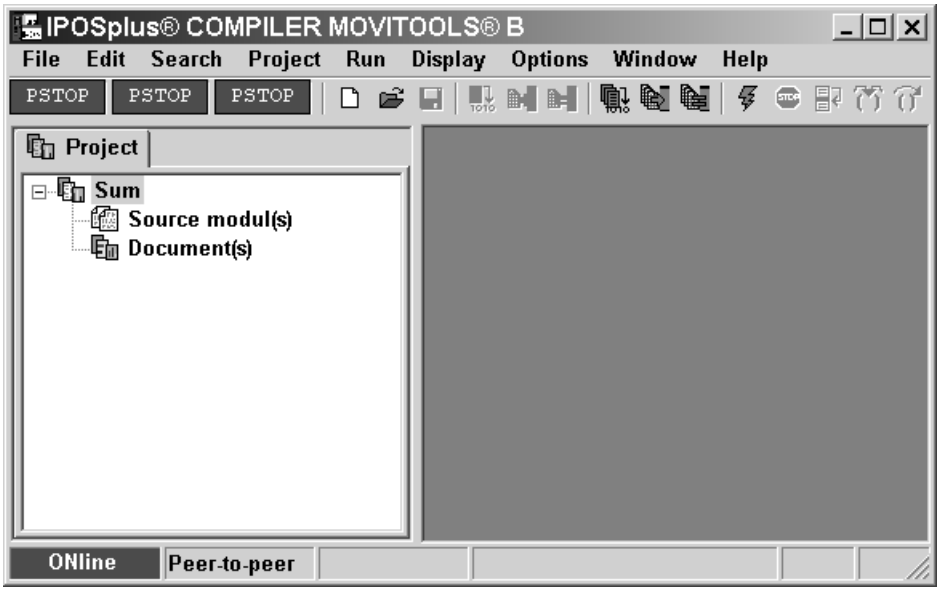
Figura 36: Janela para colocação em operação MOVITOOLS®



06072AXX

Figura 37: Busca pela pasta: Seleciona o diretório de trabalho.

A tela a seguir aparece após a interface do compilador IPOS$^{plus®}$ ter sido iniciada:



10456AEN

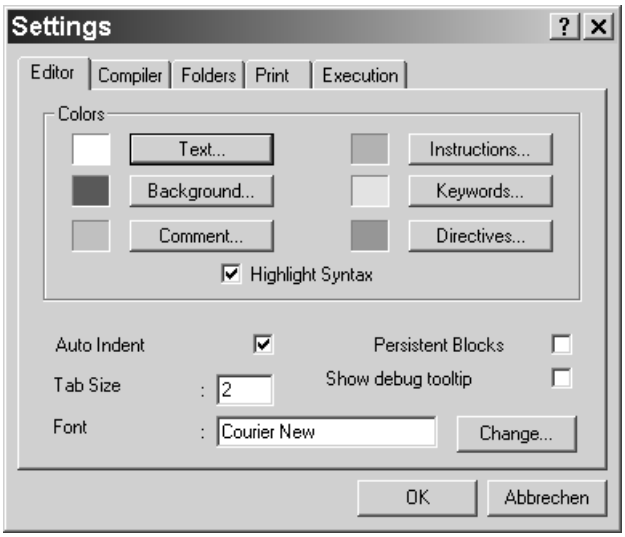
Figura 38: Interface editor para o compilador

A interface editor é dividida em quatro áreas:

- Barra do menu
- Barra de símbolos
- Janela do programa
- Barra de estado

10.4 Ajustes para o compilador IPOS$^{plus®}$

Pode-se fazer vários ajustes para o compilador inteiro. Para isto, escolher [Options] / [Settings]. Aparece a seguinte caixa de diálogo:



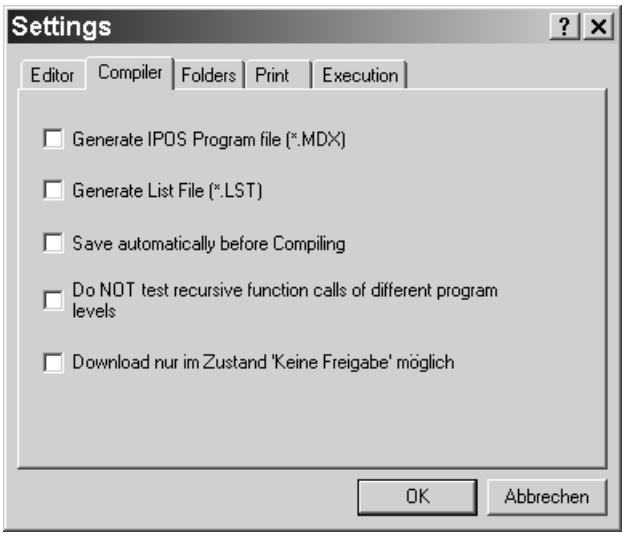
10465AEN

Figura 39: Ajustes do Editor

Você pode escolher as cores para o texto e para o fundo na página Editor. Do mesmo modo, os ajustes podem ser feitos para as cores de destaque da sintaxe, para indicar as instruções e palavras-chave a cores.

Além disso, podem ser feitos os seguintes ajustes:

- Color syntax display: Ativa e desativa o destaque da sintaxe para instruções e palavras-chave.
- Automatic indent: O cursor desloca para a direita automaticamente na linha com o primeiro caracter da linha anterior, quando mudar para uma nova linha pressionando a tecla Enter.
- Persistent blocks: Os blocos selecionados permanecem marcados até ser feita uma nova seleção. Se esta opção é desativada, a marca da seleção do bloco de texto desaparece quando o cursor é movido. Pressionar uma tecla faz com que o bloco selecionado seja substituído.
- Tab size: Número de caracteres que o cursor desloca para a direita quando a tecla Tab é pressionada.
- Font : Selecionar a fonte clicando na tecla [Change...].
- Show debug tooltip: Se a opção [Show debug tooltip] é ativada, o conteúdo das variáveis é indicado diretamente na janela Editor quando o cursor é colocado na variável desejada.

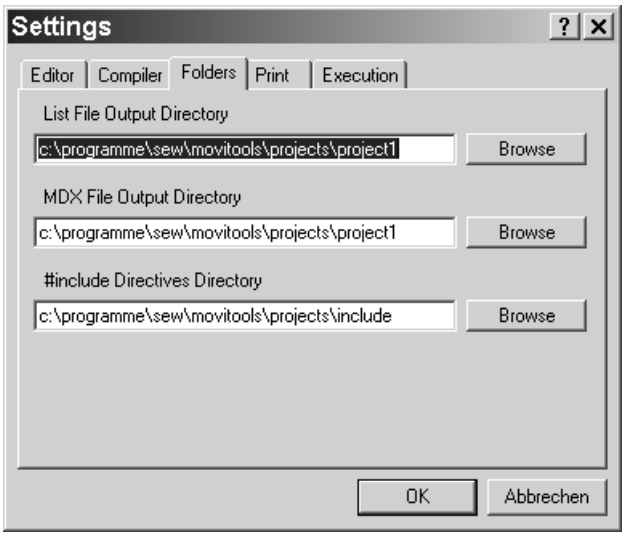


10466AEN

Figura 40: Ajustes do compilador

Pode-se fazer ajustes para o processo do compilador na página do compilador.

- Generate IPOS program file (*.MDX): É gerado um arquivo *.MDX durante a compilação. O arquivo MDX contém o código assembler do programa na forma de texto e pode ser carregado no conversor através do SHELL (Copy unit data) ou aberto e processado no assembler.
- Generate List File (*.LST): É gerada e salva uma lista do arquivo durante o processo de compilação. Ela contém informação sobre a utilização do recurso e a seqüência do programa.
- Save automatically before compiling: Se esta caixa for selecionada, o código é salvo automaticamente antes do início de cada processo de compilação.
- Do not test recursive function calls: Se esta caixa não estiver selecionada, o compilador relata um erro quando ele identifica um acesso de função repetido. Caso você queira permitir os acessos de função repetidos, você pode desabilitar esta caixa. O compilador emite apenas uma mensagem de aviso.
- Download only possible in the status "No enable": Se esta caixa estiver selecionada, o programa IPOS$^{plus®}$ somente pode ser baixado no conversor quando estiver no estado "No enable."



10467AEN

Figura 41: Ajustes do diretório

No compilador IPOS$^{plus®}$, pode ser criado um programa como projeto ou como arquivo de fonte individual.

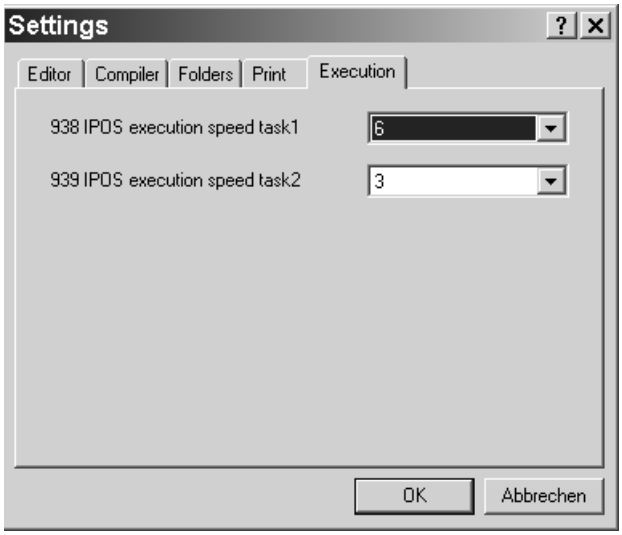
Se o programa for criado como arquivo de fonte individual, você deve fazer os seguintes ajustes, como mostrado na janela acima:

No campo [List File Output Directory], especificar a pasta a qual o arquivo deve ser salvo se esta função tiver sido ativada nos ajustes do compilador. Você pode procurar e selecionar este diretório clicando na tecla [Browse].

No campo [MDX File Output Directory], especificar a pasta a qual o arquivo MDX deve ser salvo se esta função tiver sido ativada nos ajustes do compilador. Você pode procurar e selecionar este diretório clicando na tecla [Browse].

O ajuste [#include Directives Directory] é feito no último campo nesta página. Este campo contém os detalhes do diretório o qual os arquivos de cabeçalho vinculados com o comando #include são armazenados.

Se o programa IPOS$^{plus®}$ é criado como um projeto, os ajustes na página [Directories] não são relevantes.



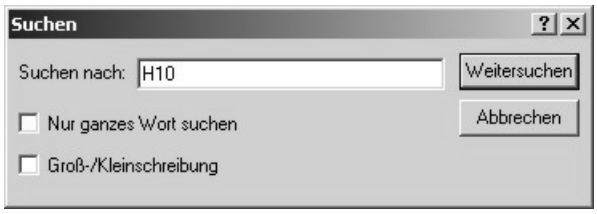
10468AEN

Figura 42: Ajuste das fases do interpretador do task

A página [Execution] pode ser utilizada para ajustar os parâmetros para a velocidade do task 1 e task 2. Estes ajustes são descritos com detalhe na seção "Gerenciamento do Task e Interrupções / Tasks para MOVIDRIVE® B."

10.5 Função de busca

Escolher [Search] / [Search for...] da barra do menu. Se você tiver marcado uma seção do texto, este texto será utilizado como a fileira de busca (na tela a seguir é "H10"). Aparece a seguinte janela:

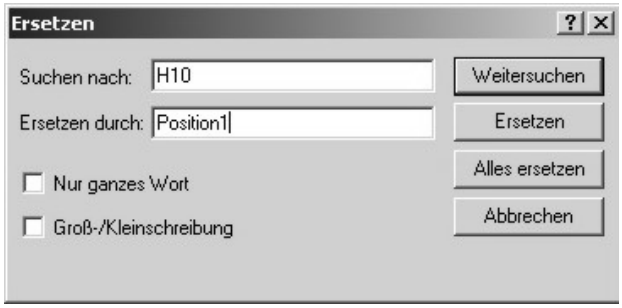


06078AXX

Figura 43: Janela [Search]

Clicar na tecla [Find Next] para buscar a próxima ocorrência da palavra em questão. Clicar [Cancel] para fechar a janela novamente.

A mesma funcionalidade também é disponível para a função [Replace...] no item [Search] na barra de menu.



06079AXX

Figura 44: Janela [Replace]

A tecla [Find Next] pode ser utilizada para procurar a palavra correspondente, que pode então ser substituída por outra palavra utilizando a função [Replace]. Clicar na tecla [Replace All] para substituir todas as combinações do termo de busca. Clicar [Cancel] para fechar esta janela.

10.6 Criando um novo projeto

Um programa IPOS$^{plus®}$ consiste de um ou mais módulos de texto fonte. Cada módulo é armazenado em um arquivo separado com a extensão *.IPC. A informação sobre o projeto é armazenada em um arquivo com a extensão *.ICP. Este arquivo digital é armazenado e administrado pelo compilador.

Define as características do projeto

Selecionar [Project] / [Create new]. Aparece uma caixa de diálogo: Inserir aqui as características gerais do projeto.

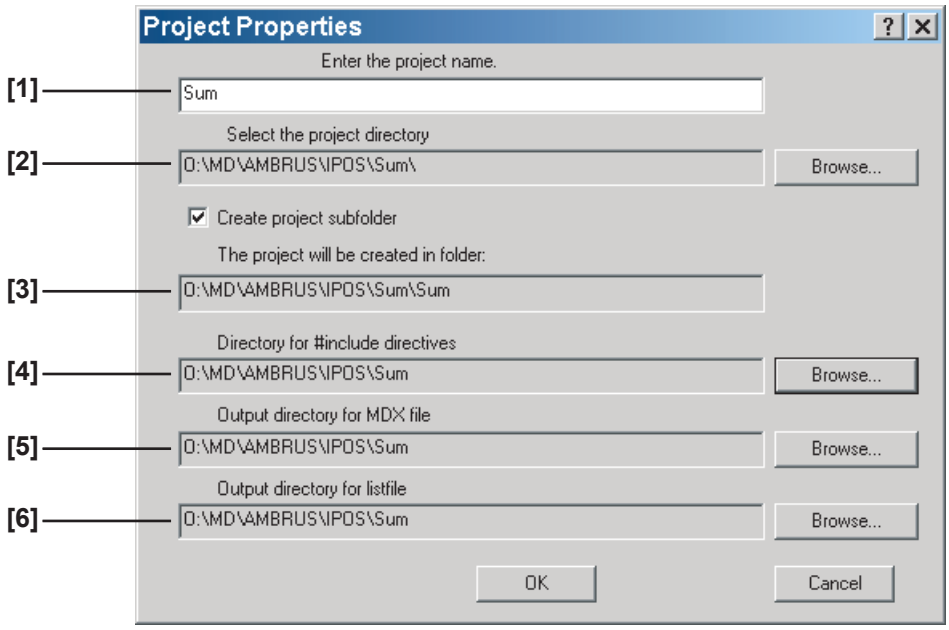


Figura 45: Características do projeto

[1] Nome do projeto
[2] Diretório do projeto
[3] Diretório o qual a pasta do projeto deve ser criada
[4] Diretório o qual os arquivos que são inseridos utilizando a indicação #include são armazenados.
[5] Diretório de saída para arquivo MDX (se ativado).
[6] Diretório de saída para arquivo de lista (se ativado).

Depois de ter confirmado suas entradas pressionando a tecla [OK], o projeto recém-criado aparece na estrutura de árvore na janela do projeto.

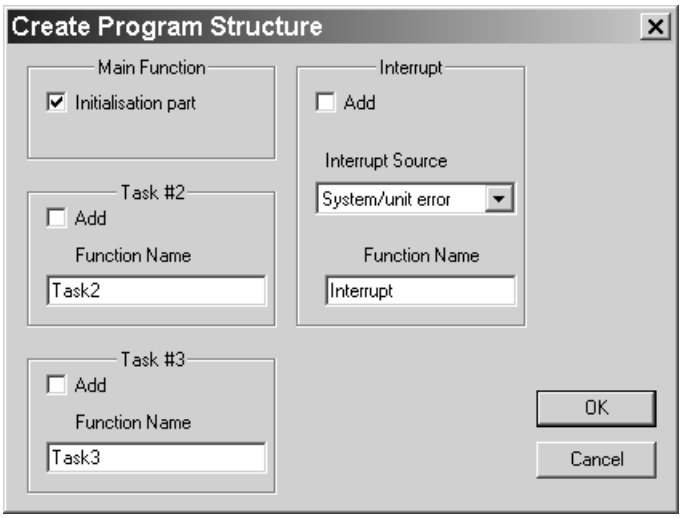
O nó principal é o nome do projeto. Os nós `Source file(s)` e `Documents` são indicados abaixo do nó principal. Os módulos fonte (*.IPC files) são indicados abaixo do `source file(s)`.

Todos os módulos fonte contidos nesta caixa são compilados para formar um programa IPOS$^{plus®}$.

Abaixo do nó `Documents`, você pode juntar qualquer arquivo necessário para a documentação (por exemplo, documentos do Word) (clicar com o botão direito do mouse: "Add document to the project'). Todos os arquivos deste nó são excluídos do processo de compilação.

Na próxima caixa de diálogo que aparece, dar um nome ao novo arquivo fonte. Se você sair da caixa selecionando [Save], abre outra caixa de diálogo.

Esta caixa de diálogo pode ser utilizada para definir uma estrutura básica do programa que é indicada como um modelo vazio de programa na janela Editor.



10458AEN

Figura 46: Definindo a estrutura do programa

Definindo a estrutura do programa

Selecionar a caixa [Initialisation part] se o programa principal tiver que conter uma parte de inicialização a qual, por exemplo, as variáveis são inicializadas.

Também, selecionar a caixa [Add] para task 2 / task 3 se for criada uma estrutura básica para task 2 / task 3. Neste caso, também é possível inserir o nome da função para task 2 / task 3 que é adotado diretamente na estrutura básica. Uma parte de inicialização é acrescentada automaticamente à estrutura básica se for adicionado o task 2 / task 3. Esta parte contém o comando para iniciar o task 2.

Se for programada uma rotina de interrupção, sua estrutura básica também pode ser criada neste estágio. Esta estrutura é criada selecionando a caixa [Add] correspondente. Utilizar o campo [Interrupt Source] para selecionar se a interrupção é por error, timer ou touch probe. O nome da função especificada é adotado na estrutura básica como o nome da função de interrupção. É inserida na parte de inicialização uma linha de indicação para ativar a rotina de interrupção.

Clicar na tecla [OK] para completar o processo de seleção da estrutura do programa. Se você clicar na tecla [Cancel] para completar o processo de seleção, o programa indica uma janela de edição vazia sem uma estrutura do programa. Isto é necessário, por exemplo, para criar seu próprio arquivo de cabeçalho.

Por exemplo, se for selecionada uma estrutura com uma parte de inicialização e um task 2, onde o task 2 tem o nome da função 'Monitoração,' a janela Editor resultante indica a seguinte estrutura do programa:

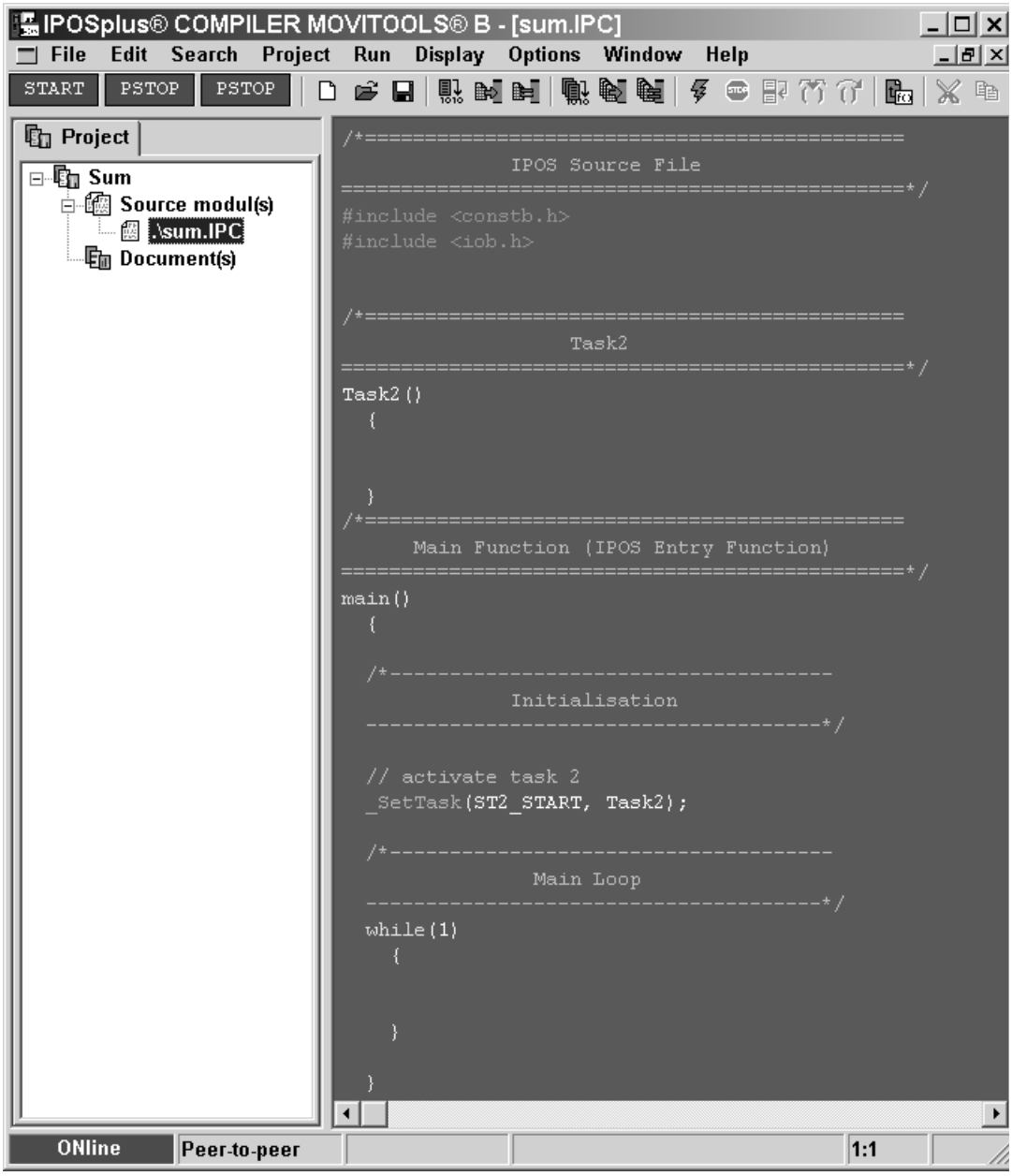
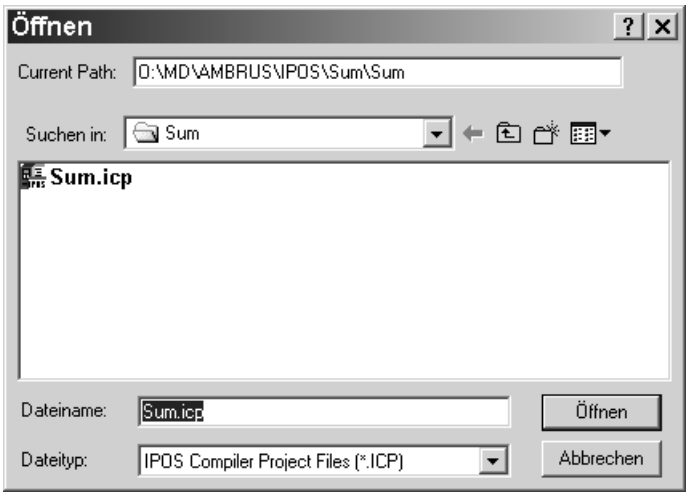


Figura 47: Programa Compilador

O compilador IPOS$^{plus®}$ é suscetível, o que quer dizer que pode ter 2 variáveis diferentes MYVAR e myvar.

Não são permitidas palavras acentuadas.

Você também tem a opção de acrescentar um arquivo fonte existente para um projeto. Para isto, clicar com o botão direito do mouse no nó principal `Source file` e selecionar [Add source file to project] do menu. Aparece a caixa de diálogo a seguir.



O tipo do arquivo é ajustado para *.ipc como padrão. Os arquivos com a terminação *.ipc indicam arquivos fonte. Os arquivos de cabeçalho com a terminação *.h também podem ser selecionados e destinados ao projeto.

Quando um arquivo é selecionado, ele aparece abaixo do nó principal do(s) arquivo(s) fonte e é destinado ao projeto.

10.7 Salvando um projeto

Há várias opções para salvar um projeto e os arquivos fonte contidos nele.

- O projeto completo e todos os seus arquivos fonte, podem ser salvos escolhendo [File] / [Save All].
- Se forem salvas somente as mudanças feitas atualmente no arquivo fonte, você pode escolher [File] / [Save] ou o ícone da barra de ferramentas.
- Escolher [File] / [Save As...] para salvar com um nome diferente o arquivo fonte ativo na janela do projeto.

10.8 Elaboração da estrutura de gerenciamento do projeto

Uma estrutura de gerenciamento do projeto permite acesso a todos os dados relativos ao projeto. Por exemplo, a estrutura a seguir mostra uma visão geral:

- Projeto principal (por ex. máquina ou cliente)
  - Documentação completa (documentos para o projeto inteiro)
  - Projeto (conversor simples)
    - Documentação (documentação individual para o conversor se não estiver armazenada na documentação completa)
    - Fonte (todos os arquivos *.IPC, todos os arquivos .h, incluindo const.h)
    - Parâmetros (arquivo *.mdx para ampliação da unidade)
    - Medidas (Arquivos do scope)
  - Projeto (conversor simples)
    - Documentação (documentação individual para o conversor se não estiver armazenada na documentação completa)
    - Fonte (todos os arquivos *.IPC, todos os arquivos .h, incluindo const.h)
    - Parâmetros (arquivo *.mdx para ampliação da unidade)
    - Medidas (Arquivos do scope)

Exemplo:

Cliente: Müller
Máquina: Estação de elevação
MOVIDRIVE®: eixo de elevação, acionamento do garfo

- Documentação completa
- Eixo de elevação
  - Documentação
  - Fonte
  - Parâmetros
  - Medidas
- Acionamento do garfo
  - Documentação
  - Fonte
  - Parâmetros
  - Medidas

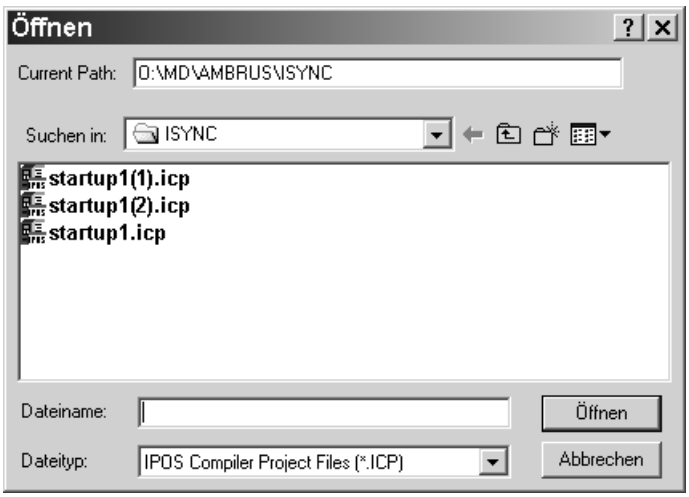
Uma estrutura de gerenciamento do projeto como esta, permite que todos se familiarizem com a máquina ou com o programa para obter um resumo rápido. A documentação e os textos fonte podem ser rapidamente localizados, tornando fácil manter um resumo. Isto facilita a manutenção do programa e do sistema todo.

As pastas e sub pastas podem ser criadas no gerenciamento do projeto no MOVITOOLS® Manager.

10.9 Abrindo um projeto

Se o compilador é aberto a partir do MOVITOOLS® Manager, o Editor é aberto com o último texto fonte a ser processado no último projeto que foi aberto, contanto que o compilador IPOS$^{plus®}$ tenha saído quando a janela Editor estava aberta.

Um projeto existente também pode ser aberto escolhendo [Project] / [Open].



Você pode buscar o arquivo de projeto na caixa de diálogo e abrí-lo utilizando a tecla [Open].

10.10 Manuseio do projeto com MOVIDRIVE® B

Com MOVIDRIVE® B, você tem a opção de armazenagem de um projeto compilável, livre de erros com todos os arquivos acompanhantes no MOVIDRIVE®.

O código completo do arquivo fonte pode ser armazenado no MOVIDRIVE® e acessado a qualquer hora.

Salvando um projeto no conversor

Escolher [Project] / [Download] para salvar o projeto atual e todos os dados acompanhantes, incluindo o próprio arquivo do projeto, no MOVIDRIVE®.

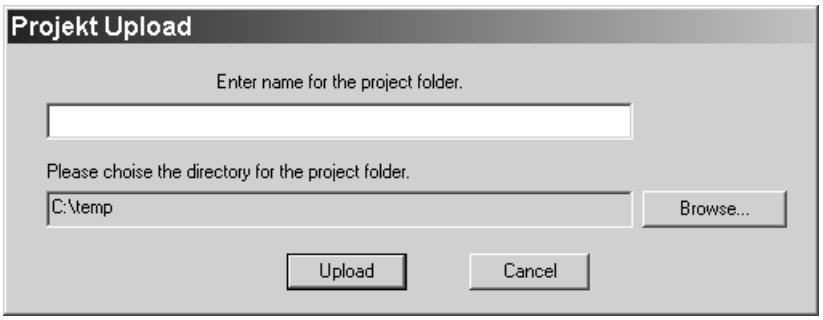
O projeto é compilado antes de ser realizado o download. Se o processo de compilação falhar, não é realizado o download dos dados.

Se o espaço disponível de memória no conversor for muito pequeno, é gerada uma mensagem de irregularidade e o processo é cancelado.

Realizando o download dos dados de projeto não realiza automaticamente o download do programa IPOS$^{plus®}$ compilado.

Carregando um projeto a partir do conversor

Escolher [Project] / [Upload...] para carregar um projeto armazenado no conversor para o PC / laptop.



Você pode inserir o nome da pasta de projeto em uma caixa de diálogo. Selecionar um diretório o qual o projeto deve ser armazenado clicando na tecla [Browse].

Se já existir no diretório um arquivo de projeto com o mesmo nome, o sistema pergunta se deve sobreescrever este arquivo.

Se não houver dados de projeto no conversor, o processo é cancelado.

Acessando um projeto a partir do conversor

Esta função é utilizada para atualizar o projeto no PC / laptop com os arquivos do conversor.

Este item do menu abre os arquivos armazenados no conversor e salva-os no diretório o qual eles se originaram, ao contrário do item do menu [Project] / [Upload].

Se já existir no diretório um arquivo com o mesmo nome, o sistema pergunta se deve sobreescrever este arquivo. Os dados de criação são inseridos para ambos arquivos para ajudar a identificá-los.

Se não houver dados de projeto disponíveis, o processo é cancelado.

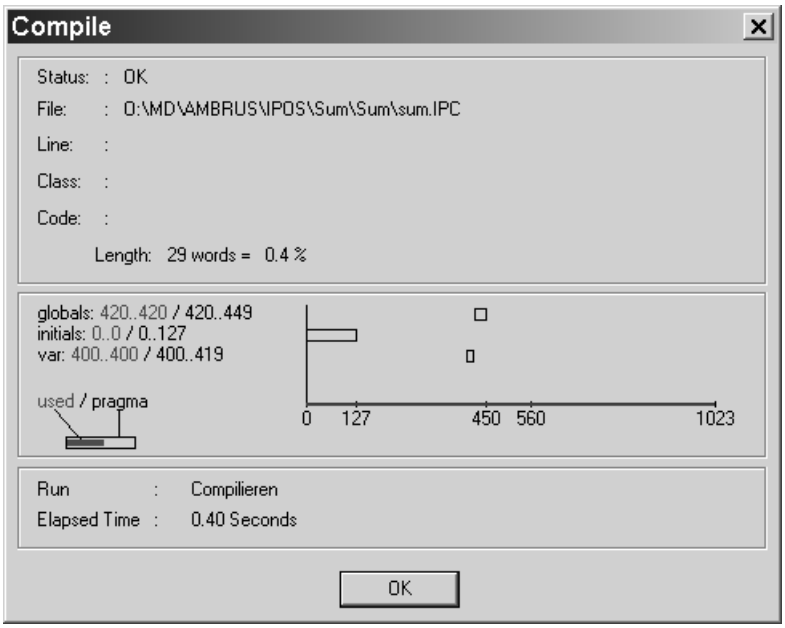
10.11 Compilando um projeto

Para gerar um programa de forma que o conversor possa entender, o projeto deve ser compilado. Se o projeto consiste de vários arquivos fonte, todos os arquivos fonte são compilados para o programa IPOS$^{plus®}$ durante o processo de compilação.

A compilação do projeto pode ser iniciada escolhendo [Project] / [Compile] ou pressionando o ícone .

O arquivo também é salvo se a função [Save automatically before Compiling] tiver sido ativada nos ajustes do compilador. Do mesmo modo, é gerado um arquivo de programa IPOS$^{plus®}$ e um arquivo de lista quando estes ajustes são ativados para o compilador.

Depois que o processo de compilação estiver completo, aparecerá a seguinte janela:



10462AEN

Figura 48: Janela de estado para compilação

- Status: Resulta do processo de compilação: OK ou ***ERROR***.
- File: Arquivo fonte do projeto o qual a irregularidade foi descoberta.
- Lines: Linhas do programa onde a irregularidade foi descoberta.
- Class: Classe de irregularidade.
- Code: Código da irregularidade.

A janela de estado também contém informação sobre o comprimento do código do programa gerado e a utilização de memória no conversor. O comprimento do código do programa é inserido como o número das palavras de código utilizado no código assembler. Este valor é utilizado para calcular e indicar a porcentagem à utilização de memória.

Pressionar a tecla [OK] para sair da janela de estado. No caso de uma irregularidade, uma barra vermelha destaca as linhas a qual a irregularidade ocorre.

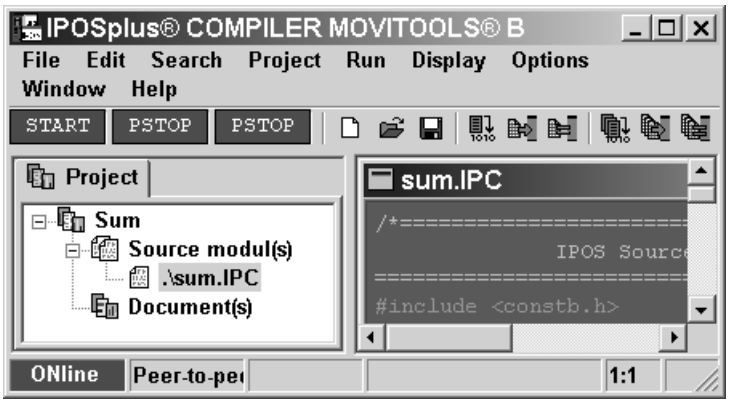
Se várias irregularidades ocorrerem em um programa, é indicada somente a primeira irregularidade na janela de estado. Depois desta irregularidade ter sido corrigida, compilar novamente o projeto e a próxima irregularidade será indicada na janela de estado.

10.12 Compilando e realizando o download

São necessárias duas fases básicas para transferir o programa para o conversor. Primeiro, o texto fonte deve ser compilado. Segundo, o programa deve ser transferido para o conversor. Disparar estas duas fases escolhendo [Project] / [Compile + download]. As condições básicas para compilação, em princípio são as mesmas daquelas descritas na seção anterior. No entanto, somente aparece uma mensagem explícita se ocorrer uma irregularidade. Você pode dizer quando a função Compile + Download foi bem sucedida porque a primeira linha do programa da função MAIN é marcada com uma barra azul clara no final do processo.

10.13 Iniciando um programa

O programa pode ser iniciado depois de ter sido feito o download para o conversor. Para fazer isto, selecionar o comando [Start] do menu [Run] na barra de menu. Por outro lado, você também pode pressionar o ícone na barra de ferramentas. A barra azul clara no editor é apagada depois do programa ter sido iniciado. O display para o estado do task muda de PSTOP para START.



10488AEN

Figura 49: Indica o estado para o task 1 e o task 2

10.14 Parando um programa

Selecionar o comando [Stop] do menu [Run] na barra de menu para parar o programa no task 1, task 2 e task 3. Por outro lado, você também pode pressionar o ícone na barra de ferramentas. O display para o estado do task na barra de ferramentas muda de START para PSTOP.

10.15 Comparar com o conversor

Há uma função de comparação do conteúdo da janela Editor com o programa no conversor. Esta função pode ser acessada selecionando o item [Compare with inverter] do menu [Project] na barra de menu. Você também pode acessar a função pressionando o ícone na barra de ferramentas.

Se as entradas combinam, é indicada a primeira janela abaixo. Se os programas não combinam, é indicada a segunda janela abaixo.

Figura 50: Programas idênticos

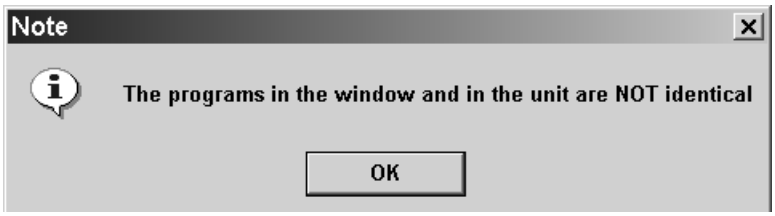


Figura 51: Programas diferentes

10.16 Debugger

O debugger integrado é uma ferramenta útil para trabalhar através de um programa de teste ou para solução de problemas em passos individuais. Para utilizar o debugger, o programa deve ser transferido para o conversor. O debugger possui três funções diferentes.

Função	Ícone	Tecla	Descrição
Execute to cursor		<F4>	O programa é processado somente até a a posição atual do cursor.
Single step		<F7>	É processada a linha do programa destacada pelo cursor. Se a função for acessada, o programa deriva desta função.
Skip		<F8>	Se as linhas do programa destacadas pelo cursor contém um acesso de função, o sistema não deriva para a função e a linha do programa é pulada.

Clicar no ícone na barra de ferramentas, pressionar a tecla de função F5 ou escolher [Run] / [Stop] da barra de menu para parar ou reiniciar o programa a qualquer hora, durante o processo de debugging.

Clicar no ícone na barra de ferramentas, pressionar a tecla de função F9 ou selecionar o comando [Start] do menu [Run] na barra de menu, para iniciar o programa da posição atual do cursor a qualquer hora, durante o debugging.

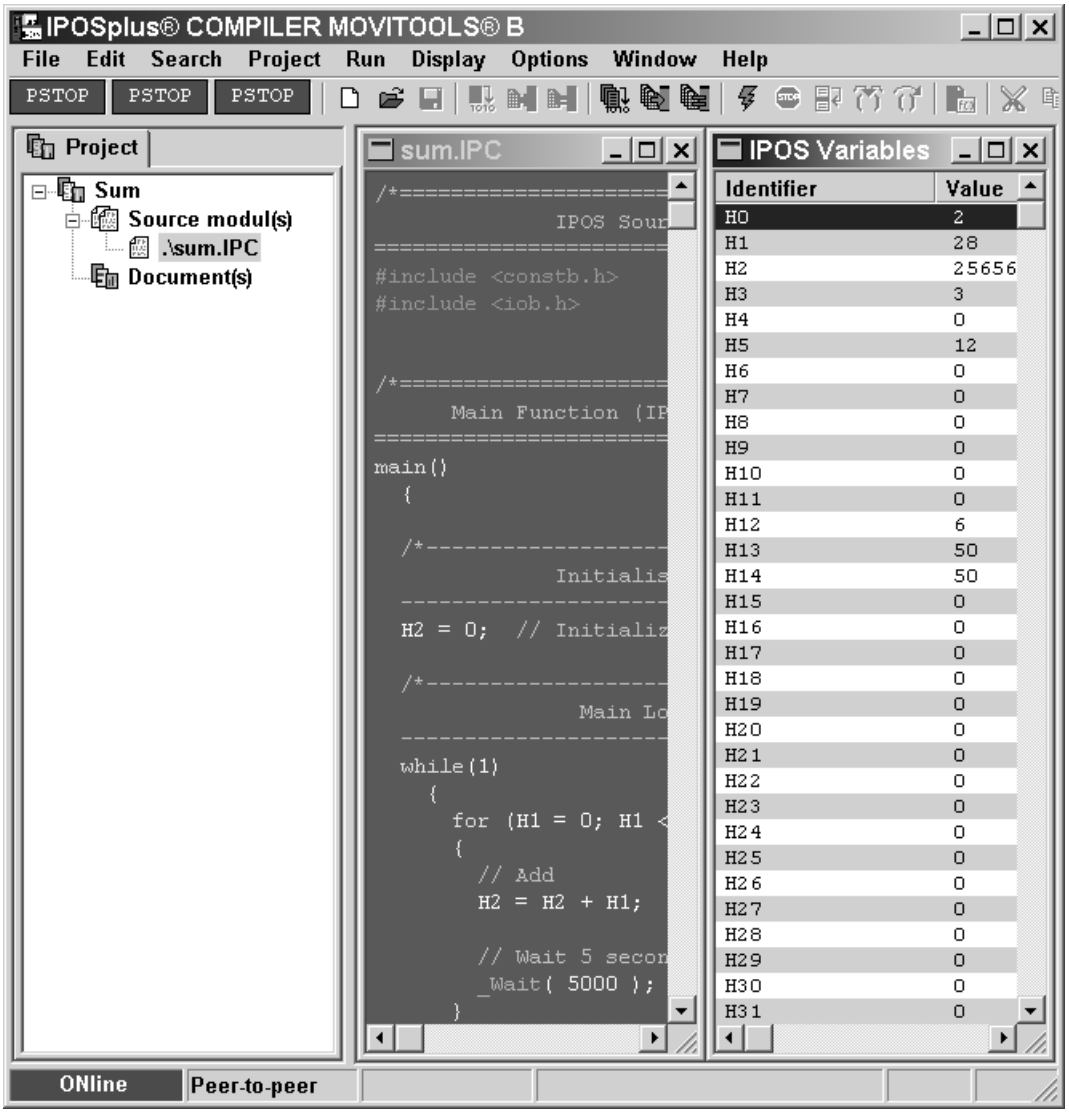
Enquanto o programa está funcionando, você pode interrompê-lo pressionando a combinação das teclas Alt+F5 ou escolhendo [Run] / [Break]. A barra de execução agora é posicionada no comando que deve ser executado a seguir.

O programa também pode ser interrompido pressionando a tecla F4 ou escolhendo [Goto cursor]. O programa é parado na linha do comando a qual o cursor é posicionado.

10.17 Janela de variáveis

É útil abrir uma janela de variáveis para que possa ser observado o conteúdo das variáveis durante o processo de debugging ou funcionamento normal do programa.

A janela de variáveis é acessada selecionando [All Variables] do menu [Display] na barra de menu.



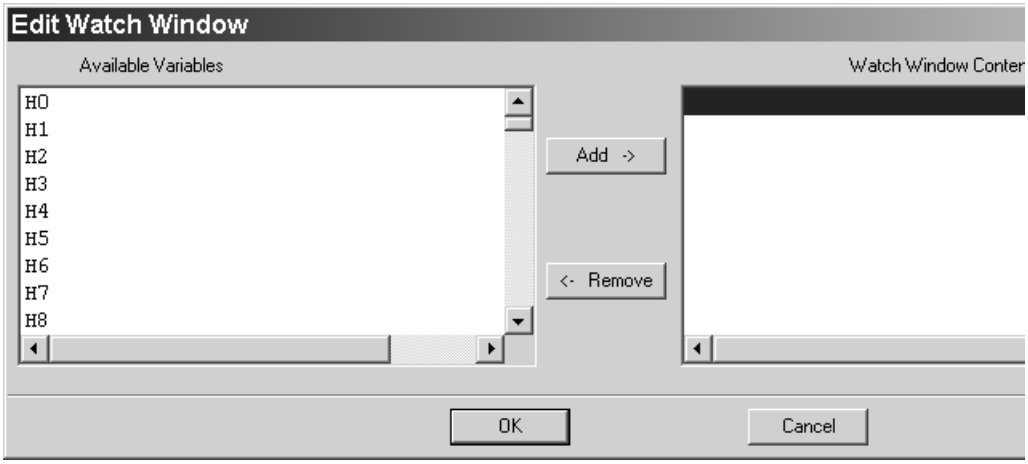
10501AEN

Figura 52: Indicando as variáveis

Todas as variáveis podem ser observadas na janela de variáveis utilizando a barra de rolagem.

Outra maneira de observar as variáveis é configurar a watch window. Na watch window somente são indicadas as variáveis selecionadas. Quanto menos variáveis indicadas ao mesmo tempo, mais rápido o valor individual pode ser atualizado.

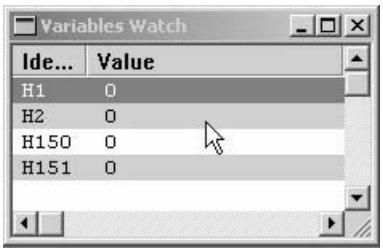
Para configurar a watch window, escolher [Display] / [Variable Watch] / [Edit Window...] da barra de menu. Aparece a seguinte janela:



10502AEN

Figura 53: Selecionando as variáveis

Clicar duas vezes na variável que você quer indicar, ou selecionar várias variáveis e pressionar [Add] para fixar as variáveis desejadas na watch window. As variáveis selecionadas são indicadas na lista à direita da janela. Para remover uma variável da watch window, destacá-la na lista e clicar na tecla [Remove].



06090AXX

Figura 54: Indica a variável selecionada

A janela parece a mesma que a janela de variáveis completa, mas contém somente as variáveis selecionadas.

As variáveis podem ter identificadores simbólicos atribuídos a elas porque os nomes padrão da variável (por ex. H1, H2, etc.) são difíceis de interpretar em programas grandes. Estes identificadores também são mostrados neste momento.

Os valores das variáveis podem ser indicados em formatos diferentes. Pode-se selecionar entre os seguintes formatos: signed decimal, hexadecimal, binary ou ASCII. Para mudar de um formato ao outro, selecionar primeiro a variável desejada clicando-a. Depois acessar o menu pressionando o botão direito do mouse, e selecionar de lá o formato desejado.

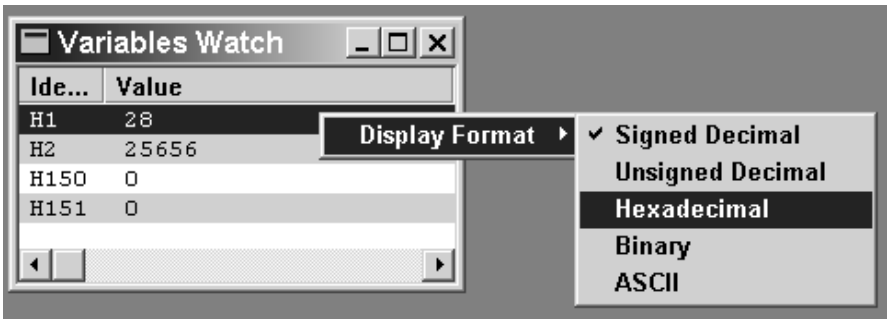
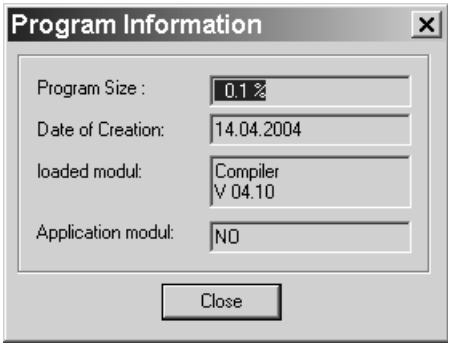


Figura 55: Mudando o formato da variável

As variáveis individuais também podem ocorrer várias vezes na watch window. Deste modo, a variável pode ser indicada em vários formatos ao mesmo tempo.

10.18 Informação do programa

O comando [Program Information] é disponível no menu [Display] na barra de menu. Se você selecionar este item do menu, aparece a seguinte janela:



10504AEN

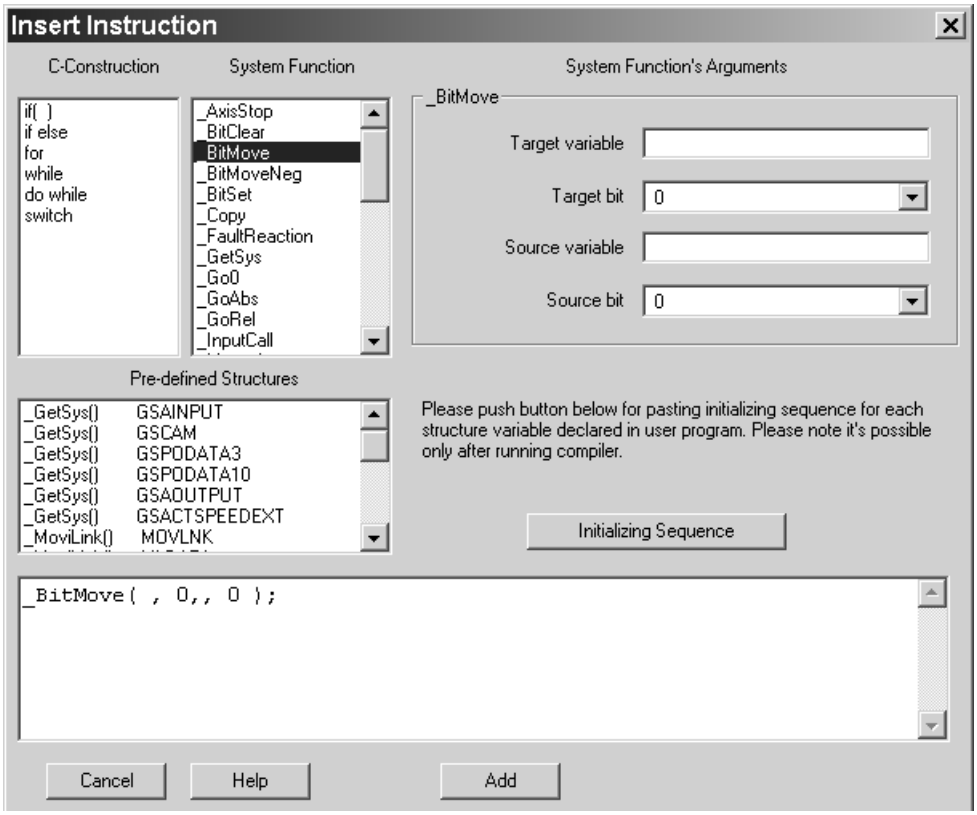
Figura 56: Informação do programa

Esta informação do programa refere-se ao programa armazenado no conversor. O tamanho do programa, os dados de criação e o nome do arquivo fonte são indicados nesta janela. Clicar na tecla [Open File] para indicar o código fonte para o programa no conversor em uma janela Editor. Isto presume que o nome do arquivo fonte não foi mudado e pode ser localizado no caminho que foi utilizado para transferir o programa ao conversor.

10.19 Inserindo comandos

Em princípio é possível inserir o texto fonte inteiro de um programa IPOS$^{plus®}$ digitando-o utilizando o teclado. Neste caso, utilizar a sintaxe baseada na linguagem de programação C. Você pode desfazer as últimas cinco entradas utilizando a combinação das teclas [Ctrl + Z].

Pode-se utilizar a ferramenta de inserção enquanto estiver editando o programa. Esta ferramenta pode ser acessada pressionando o botão direito do mouse para abrir o menu e então selecionar [Insert Instruction]. Pode-se acessar a ferramenta de inserção pressionando o ícone na barra de ferramentas.



10505AEN

Figura 57: Inserindo um comando

Pode-se utilizar a ferramenta de inserção para acrescentar as construções C, funções do sistema ou estruturas pré-definidas para o texto fonte.

Se você marcar a construção C ou uma estrutura padrão, o texto que será inserido no texto fonte aparece na parte inferior da janela. Caso queira inserir uma função do sistema, você deve inserir também os argumentos da função no lado direito da janela. Clicar na tecla [Add] para inserir a função selecionada no texto onde o cursor foi posicionado quando você acessou a ferramenta de inserção.

Uma parte da inicialização é acrescentada em cada estrutura variável definida quando você clica [Initializing Sequence]. Para fazer isto, o programa deve ter sido compilado pelo menos uma vez.

10.20 Comentários

O bom uso dos comentários facilita a leitura do programa e também torna possível para quem não conhece o programa, familiarizar-se rapidamente com ele.

Um comentário longo, que pode ocupar várias linhas, por exemplo, inicia com '/*' e termina com '*/'. Um comentário de uma linha inicia com '//' e não precisa de uma marca final. Um comentário de uma linha também pode ser inserido diretamente após uma linha de comando no texto fonte.

10460AEN

Figura 58: Comentários

10.21 Resumo dos ícones

Ícone	Item do menu	Descrição
	File → New	Cria novo arquivo fonte
	File → Open	Abre arquivo fonte
	File → Save	Salva arquivo fonte
	File → Compile	Compila arquivo fonte
	File → Compile + download	Compila arquivo fonte e realiza o download para o conversor
	File → Compare with inverter	Compara o arquivo fonte com o programa no conversor
	Project → Compile	Compila o projeto
	Project → Compile + download	Compila o projeto e realiza o download para o conversor
	Project → Compare with inverter	Compara o projeto com o programa no conversor
	Run → Start	Inicia o programa IPOS$^{plus®}$
	Run → Stop	Pára o programa IPOS$^{plus®}$
	Run → Run to cursor	Roda o programa para onde o cursor está posicionado
	Run → Single step	Roda um passo
	Run → Skip	Pula uma instrução (comando)
	Edit → Insert instruction	Acessa a ferramenta de inserção
	File → Print	Imprime o arquivo fonte
	Help → User manual	Acessa a ajuda online

11 Compiler – Programação

O texto fonte de um programa escrito com o compilador IPOS$^{plus®}$ é constituído de várias partes. Estas partes devem ser consideradas primeiro individualmente.

```
/*********************************************
File name:          Program_structure.IPC
Date: 04.02.2002
Author: Thomas Ambrus
SEW-EURODRIVE Bruchsal
Technical Documentation
Brief description:  Source code program structure
*********************************************/

/*============================================
IPOS Source File
============================================*/
#include <const.h>
#include <io.h>

/*============================================
Main Function (IPOS Entry Function)
============================================*/
main()
{

  /*-----------------------------------
  Initialization
  -----------------------------------*/

  // Activate task 2
  _SetTask2(T2_START, Monitor); // testing

  /*-----------------------------------
  Main loop
  -----------------------------------*/
  while(1)
  {

  }
}

/*============================================
Task2
============================================*/
Monitor()
{

}

/*============================================
User function
============================================*/
Reference travel()
{

}

Automatic mode()
{

}

Manual mode()
{

}
```

Comentário com observações no programa

Cabeçalho do programa com comandos pré-processadores e, se necessário, definição das variáveis

A função principal contém parte da inicialização e o circuito contínuo para task 1

Parte da inicialização

Circuito contínuo para task 1

Task 2, não é necessário circuito contínuo

Funções (subprogramas) criadas pelo usuário, acessadas do task 1 e task 2

11.1 Pré-processador

O compilador IPOS$^{plus®}$ é um compilador multi-pass que processa o texto fonte em várias voltas completas. Durante a primeira volta completa, o pré-processador processa os comandos – referido abaixo como diretivas – que são destinadas a isto, testa os comandos para compilação condicional, apaga os comentários e cria finalmente um arquivo temporário para o compilador. O pré-processador aumenta a flexibilidade e a produtividade durante a programação nas seguintes áreas:

- Integração do texto a partir de outros arquivos (arquivos de cabeçalho) que contém constantes preparadas e/ou definidas pelo usuário ou funções de texto fonte.
- Definição dos identificadores simbólicos para melhorar a legibilidade do texto fonte.
- Definição das diretivas para compilação condicional para melhorar a portabilidade e simplificar as fases de teste.

Cada linha iniciando com # é tratada como uma diretiva do pré-processador, a menos que # seja parte de um comentário. Qualquer espaço em branco antes ou depois do caracter # é ignorado.

As diretivas do pré-processador geralmente são escritas no início do texto fonte. Elas podem, no entanto, estar localizadas em qualquer lugar no programa. Dependendo da função das diretivas, elas são aplicáveis para a linha do texto fonte a qual elas estão localizadas ou para o programa inteiro independente das suas localizações.

11.2 Comandos do pré-processador

As linhas de comentário no cabeçalho do programa são seguidas pelos comandos do pré-processador. Um comando deste tipo é inserido como padrão quando você abre uma nova janela Editor.

O comando '#include <const.h>' integra um arquivo de cabeçalho chamado const.h quando o texto fonte é compilado. Este arquivo tem um formato fixo e não deve ser modificado. Todavia, nós explicaremos a função de um arquivo de cabeçalho com relação a este arquivo. Uma forma abreviada do arquivo é impressa abaixo porque é suficiente para demonstrar os aspectos principais.

```
/*============================================
File name: Const.h
File version: 2.20

SEW Include-File for IPOSplus Compiler

Please do not modify this file!

(C) 1999 SEW-EURODRIVE
============================================*/

#ifndef _CONST_H
#define _CONST_H
.
.
.
#define Scope474    H474
#define Scope475 H475
#define DRS_Ctrl H476
#define DRS_Status H477
#define AnaOutIPOS2 H478
#define AnaOutpIPOS H479
#define OptOutpIPOS H480
#define StdOutpIPOS H481
#define OutputLevel H482
#define InputLevel H483
#define ControlWord H484
#define T0_Reload   H485
#define Reserve4    H486
#define Timer_2 H487
#define Timer_1 H488
#define Timer_0 H489
#define WdogTimer   H490
#define SetpointPos H491
#define TargetPos H492
#define PosWindow   H493
#define LagWindow   H494
#define LagDistance H495
#define SLS_right H496
#define SLS_left H497
#define RefOffset   H498
#define SetpPosBus  H499
#define Reserve6 H500
#define Reserve7    H501
#define TpPos2_Abs  H502
#define TpPos1_Abs  H503
#define TpPos2_Ext  H504
#define TpPos2_Mot  H505
#define TpPos1_Ext  H506
#define TpPos1_Mot  H507
#define Reserve8    H508
#define ActPos_Abs H509
#define ActPos_Ext H510
#define ActPos_Mot H511

#endif
```

A estrutura atual do arquivo de cabeçalho inicia com '#ifndef _CONST_H' após uma seção de comentário geral. Este comando '#ifndef' é sempre acompanhado por '#endif'. Você pode encontrar este comando '#endif' na última linha do programa do arquivo de cabeçalho. Esta construção '#ifndef' e '#endif' é utilizada para evitar que o arquivo seja vinculado mais do que uma vez.

Os comandos dentro desta construção somente são realizados se um identificador macro, aqui '_CONST_H,' ainda não tiver sido definido (**se n**ão **def**inido). A '#define _CONST_H' é posicionada na próxima linha do programa para definir este identificador macro. Portanto, se o arquivo de cabeçalho é processado durante a compilação pelo comando '#include <const.h>', a pergunta '#ifndef _CONST_H' é respondida inicialmente na afirmativa porque o identificador macro '_CONST_H' ainda não está identificado. Então ele é definido com '#define _CONST_H.' Se o arquivo de cabeçalho 'const.h' é vinculado em outro lugar no programa, o identificador macro '_CONST_H' já foi identificado e a pergunta '#ifndef _CONST_H' é negada. Como resultado, o processamento pula imediatamente para o comando '#endif'. Isto evita que o arquivo seja incorporado mais do que uma vez desnecessariamente, o qual conduziria a uma mensagem de erro.

Além do comando '#ifndef', há também o comando '#ifdef' (**se def**inido). Este comando não deve ser negado. Uma construção if-else também é possível. Neste caso, isto significa que a parte do comando seguinte do '#else' é processada se a pergunta '#ifdef' ou '#ifndef' não for realizada. Isto resulta nas seguintes possibilidades:

#ifdef identifier_1 Program text_1 #else Program text_2 #endif	#ifndef identifier_2 Program text_3 #else Program text_4 #endif	#ifdef identifier_3 Program text_5 #endif

Observe que estes comandos do pré-processador também podem ser utilizados para um efeito bom no programa principal, não apenas nos arquivos de cabeçalho. Como resultado, por exemplo, as partes de um programa podem ser convertidas especificamente para uma máquina, sem ter que fazer mudanças maiores no texto fonte.

11.3 #include

Esta diretiva possibilita incorporar textos fonte de outros arquivos (arquivos de cabeçalho) no arquivo texto fonte. Os arquivos de cabeçalho são utilizados normalmente para definir constantes ou macros que são utilizadas várias vezes, assim estão disponíveis em projetos diferentes. A sintaxe é:

```
#include <FileName>
```

`FileName` é o nome completo do arquivo a ser incorporado. Ele é incluso dentro dos sinais < e >. É suficiente indicar o nome do arquivo sem a informação do caminho se o arquivo a ser incorporado estiver localizado na pasta atual

O arquivo `BEISPIEL.IPC` contém o programa principal.

```
#include <CONST.H>
H10 = MAXIMUM_SPEED;
```

O pré-processador substitui a diretiva `#include` com o conteúdo do arquivo `CONST.H`:

```
#define MAXIMUM_SPEED 3000
H10 = MAXIMUM_SPEED
```

O resultado após a expansão da macro é o seguinte:

```
H10 = 3000;
```

O arquivo `CONST.H` é um arquivo de cabeçalho.

```
#define MAXIMUM_SPEED 3000
```

As diretivas #include também podem ser utilizadas em estruturas aninhadas, isto é, um arquivo incorporado pode por si só conter uma diretiva #include para incorporar outro arquivo. Certifique-se de que os arquivos não montem uma rotina de inclusão (eles incluem-se por si só). Isto leva a um erro do pré-processador. Recomendamos evitar aninhamento das diretivas #include para manter a estrutura limpa.

11.4 Incluir pastas

Há vários procedimentos dependendo da pasta a qual o arquivo a ser incluso é localizado.

1. Se o caminho do arquivo a ser incorporado é ajustado em Folders tab page nos ajustes do compilador, então o comando é #include <FileName> onde FileName é o nome do arquivo de cabeçalho.
2. Se o arquivo a ser incorporado está localizado na pasta de trabalho atual, então o comando é #include 'FileName.' FileName é o nome do arquivo a ser incorporado.
3. O caminho da pasta deve ser especificado se o arquivo a ser incorporado estiver localizado em uma outra pasta diferente daquelas já constatadas aqui. Por exemplo, o comando para incorporação de um arquivo chamado Test.h localizado na pasta principal seria '#include 'c:\Test.h''.

O melhor lugar para colocar arquivos de cabeçalho que você mesmo escreveu é na pasta de trabalho atual. Isto permite que o programa seja escrito independente do caminho da pasta. Não há necessidade de fazer mudanças se o programa for compilado em uma pasta diferente e a estrutura do diretório do programa for mantida. O programa pode ser compilado imediatamente.

O ajuste para o compilador teria que ser mudado no primeiro método, enquanto o caminho #include teria que ser editado no programa no terceiro método antes do programa ser re-compilado.

O sistema busca na seqüência a seguir, se todos os três métodos estão combinados:

1. Programação do caminho direto no comando #include (método 3)
2. Programação do caminho relativo ao arquivo fonte (método 2)
3. Programação do caminho na diretiva #include dos ajustes da caixa de diálogo

11.5 #define

A diretiva '#define' foi utilizada anteriormente para definir um identificador macro. No entanto, a função básica da diretiva '#define' é definir uma macro. As macros são utilizadas para substituir símbolos no texto fonte por caracteres. Este mecanismo possibilita formular simbolicamente constantes, variáveis, etc.. O compilador somente realiza macros sem parâmetros. A sintaxe é:

```
#define MacroIdentifier   <SymbolSequence>
```

Cada ocorrência de uma 'MacroIdentifier' no texto fonte seguindo esta diretiva é substituído por 'SymbolSequence' (que pode estar vazio). A MacroIdentifier é definida se o SymbolSequence estiver vazio; não tem nenhuma outra função. A seqüência do símbolo não deve exceder 75 caracteres.

Deste modo, uma notação simbólica é atribuída às variáveis do sistema no arquivo de cabeçalho. Portanto, por exemplo, a variável H474 pode ser endereçada utilizando o nome simbólico 'Scope474' ou a variável H484 utilizando 'ControlWord' após o arquivo de cabeçalho const.h ter sido incorporado.

Igualmente, a diretiva '#define' pode ser utilizada para atribuir nomes simbólicos aos valores constantes. Como resultado, '#define MAX_SPEED 1500' possibilita a 'MAX_SPEED' ser escrita no texto fonte melhor do que a figura específica de 1500. Isto facilita a leitura do texto fonte.

O exemplo a seguir ilustra este ponto:

```
#define setpoint    H123
#define maximum     2000
setpoint = maximum; // in this line the macro definitions 'setpoint' and 'maximum'
                    // are replaced, meaning: H123 = 2000;
```

Após cada expansão da macro, o texto resultante é novamente examinado. Isto possibilita utilizar macros aninhadas.

```
#define setpoint     H10
#define variable1    setpoint
#define minimum 20+H11
variable1 = minimum; // in this line the macro definition 'variable1' is
                     // replaced by 'setpoint', then 'setpoint' is replaced by 'H10',
                     // meaning: H10=20+H11;
```

Certifique-se de que uma variável identificada com #define **não** tenha atribuído duas variáveis do sistema por engano.

O processo de compilação não pode detectar se a variável é definida com o mesmo nome da estrutura. O conversor gera o erro 10 IPOS-ILLOP.

Exemplo:

```
#define Position H2
VARINT Position;
```

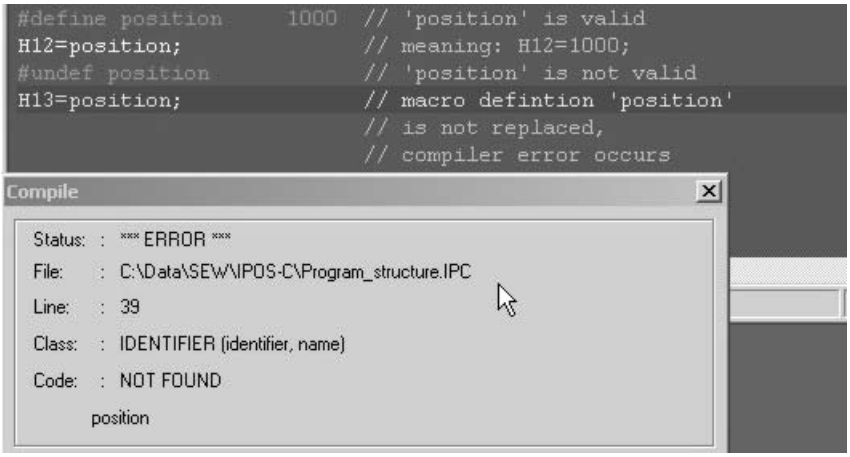
11.6 #undef

Esta diretiva possibilita desativar uma macro que foi criada anteriormente utilizando '#define...'.

```
Syntax: #undef MacroIdentifier
```

O exemplo a seguir ilustra este ponto:

```
#define position     1000  // 'position' is valid
H12=position;              // meaning: H12=1000;
#undef position // 'position' is not valid
H13=position; // macro definition 'position' is not replaced,
                           // compiler error occurs
```


#define position 1000 // 'position' is valid
H12=position; // meaning: H12=1000;
#undef position // 'position' is not valid
H13=position; // macro defintion 'position'
// is not replaced,
// compiler error occurs
Compile
Status : *** ERROR ***
File : C:\Data\SEW\IPOS-C\Program_structure.IPC
Line : 39
Class : IDENTIFIER (identifier, name)
Code : NOT FOUND
position

06095AXX

Figura 59: Erro no compilador devido a definição faltante

11.7 #declare

Esta diretiva permite que as variáveis IPOS$^{plus®}$ sejam declaradas simbolicamente e relativas a uma variável base. Isto facilita a portabilidade dos módulos de texto fonte tanto quanto a atribuição dos números variáveis, porque o usuário somente tem que mudar o número da variável base para mudar todos os números variáveis utilizados no texto fonte.

Deste modo, é mais fácil integrar módulos pré-configurados em seu próprio texto fonte contanto que estes módulos tenham números variáveis relativos.

`Syntax: #declare IdentifierNew  IdentifierOld : Offset`

O exemplo a seguir ilustra esta função:

```
#define basevariable  H100

#declare setpoint     basevariable:0
#declare actvalue     basevariable:1
#declare i            basevariable:5
```

As variáveis a seguir são agora disponíveis como variáveis simbólicas: setpoint, actual value e i. Além disso, isto especifica também que as variáveis IPOS$^{plus®}$ H100, H101 e H105 são atribuídas.

Podem ser utilizadas no máximo 600 diretivas #define e #declare.

O task dos módulos de integração é feito mais fácil pela formação dos blocos da variável utilizando diretivas de declaração. No entanto, continua muito difícil tratar disso porque o usuário precisa ter uma visão geral das variáveis que estão ocupadas e das que ainda estão disponíveis. Portanto, é uma boa idéia utilizar estruturas, estruturas padrão SEW ou estruturas definidas pelo usuário especialmente quando tem que ser fornecida uma seqüência fixa de variáveis (por ex. SETSYS, GETSYS, MOVLNK, e assim por diante). Todas as outras variáveis devem ser declaradas com as palavras-chave maiúsculas ou a inicial maiúscula como descrito abaixo, desde que isto deixe o task de atribuição dos números da variável acima do compilador.

11.8 Estruturas padrão SEW

As estruturas padrão SEW fornecem estruturas prontas para comandos que dependem de estruturas.

A tabela a seguir mostra uma lista de estruturas padrão disponíveis para cada comando específico, junto com seus elementos correspondentes.

Comando	Estrutura padrão	Elementos	Descrição breve
_GetSys	GSAINPUT	Input1	Valor de tensão da entrada analógica 1
		Input2	Valor de tensão da entrada analógica 2
	GSAOUTPUT	Output1	Valor de tensão para saída analógica opcional 1
		Output2	Valor de tensão para saída analógica opcional 2
	GSCAM	SourceVar	Número da variável a qual o comando é realizado
		DbPreCtrl	Pré-controle do tempo de atraso em 0.1 ms
		DestVar	Número da variável que é para receber o resultado
		BitPosition	Posição do bit na variável de resultado
		BitValue	Polaridade na variável de resultado
		NumOfCam	Número de blocos came (máx. 4)
		PosL1	Valor limite antihorário do bloco came 1
		PosR1	Valor limite horário do bloco came 1
		PosL2	Valor limite antihorário do bloco came 2
		PosR2	Valor limite horário do bloco came 2
		PosL3	Valor limite antihorário do bloco came 3
		PosR3	Valor limite horário do bloco came 3
		PosL4	Valor limite antihorário do bloco came 4
		PosR4	Valor limite horário do bloco came 4
	GSCAM_EXT	CamControl	O bit 2^{31} sempre deve ser ajustado. 0x8000 0000 = função inativa, nenhuma saída de came nova será gerada, o ajuste das saídas será retido e apagado somente após um reset ou quando a tensão é desligada/ligada. 0x8000 0001 = função ativa internamente, mas todas as saídas de came são desligadas 0x8000 0002 = função ativa se o conversor é referenciado (H473, Bit20 =1) 0x8000 0003 = função ativa mesmo sem o conversor referenciado
		CamReserved1	Reservado
		CamOutShiftLeft	Desloca o buffer de dados interno das saídas por n dígitos à esquerda antes de escrever na variável final CamDestination. Importante: O processo de deslocamento apaga a informação das saídas superiores. Isto significa que se o fator de deslocamento for 3, as 3 saídas superiores com duração do ciclo de 4 ms não são mais disponíveis, e as 4 saídas com duração do ciclo de 1 ms são atribuídas aos bits 3-6 e a saída com duração do ciclo de 4 ms é atribuída ao bit 7.
		CamForceOn	Máscara para ajustar as saídas obrigatórias; a máscara atinge o buffer de dados interno antes de deslocar com CamOutShiftLeft (NÃO a variável final definida com CamDestination)
		CamForceOff	Máscara para apagar as saídas obrigatórias; a máscara atinge o buffer de dados interno antes de deslocar com CamOutShiftLeft (NÃO a variável final definida com CamDestination) CamForceOff domina CamForceOn

Comando	Estrutura padrão	Elementos	Descrição breve
		CamSource	Bit 2^{31} comuta entre as variáveis de referência pré-definidas e um indicador para a variável de referência aleatória. Bit 2^{31} = 0: • 0 = encoder X15 (encoder do motor, H511) • 1 = encoder X14 (encoder externo, H510) • 2 = encoder H509 (encoder absoluto DIP11A) • 3 = encoder virtual • todos os valores seguintes são reservados! Bit 2^{31} = 1: CamSource inclui um indicador para uma variável IPOS$^{plus®}$ $+2^{31}$
		CamDestination	Indicador para a variável final. Os bits não utilizados nas variáveis finais são disponíveis para outras funções (se deslocar as saídas por quatro à esquerda com Shift Left, isto libera até os bits 0-3, os bits 4-7 são disponíveis para as funções came e os bits 8-31 são disponíveis para qualquer programação. Se as saídas came são atribuídas às saídas do conversor (por ex. H481), você tem que reservar estas saídas digitais com P620 – P639 como saídas IPOS$^{plus®}$. Os bits não utilizados nesta palavra são disponíveis para outras saídas.
		CamOutputs	Número de saídas (máx. 8)
		CamData 1	Indicador para primeira estrutura CamOutput (1ª saída)
		...	
		CamData 8	Indicador para última estrutura CamOutput (8ª saída)
	CAM_EXT_OUT	DeadTime	Compensação do tempo de atraso para este canal (–500 ms..0..+500 ms) para compensar o tempo de atraso de um atuador conectado ao conversor. A saída é pré-definida dependendo da taxa de mudança do valor da variável de referência, de tal modo que a saída é comutada com antecedência por este intervalo de tempo.
		CamAreas	Número das janelas de posição para este canal (1 ... 4); o valor limite antihorário sempre deve ser menor do que o horário. Se um módulo de eixo necessita uma janela de posição que exceda o limite 360° - 0°, então esta faixa terá que ser dividida em duas janelas de posição. Este processo deixa o operador ajustar três faixas relacionadas a esta saída.
		LeftLimit1	Limite antihorário, janela 1
		RightLimit1	Limite horário, janela 1
		...	...
		LeftLimit4	Limite antihorário, janela 4
		RightLimit4	Limite horário, janela 4
	GSPODATA3	Bus types	0 = Reservado 1 = S0 (RS485 #1) 2 = S1 (RS485 #2) 3 = Fieldbus 4 = Reservado 5 = SBus
		Len	Número de itens dos dados de saída de processo
		PO1	Dados de saída de processo 1
		PO2	Dados de saída de processo 2
		PO3	Dados de saída de processo 3

Comando	Estrutura padrão	Elementos	Descrição breve
	GSACTSPEEDEXT	TimeBase	Duração do ciclo para descoberta da rotação do encoder externo, faixa de ajuste: 5 ms ... 31 ms
		EncType	0 = encoder X14,1 = Encoder DIP
		Numerator	Numerador para escala do usuário Faixa de valor: -2^{15} ... 0 ... $+(2^{15}-1)$
		Denominator	Denominador para escala do usuário Faixa de valor: 1 ... $(2^{15}-1)$
		DPointer	Indicador para a variável de resultado H'
	GSPODATA10	Bus types	0 = Reservado 1 = S0 (RS485 #1) 2 = S1 (RS485 #2) 3 = Fieldbus 4 = Reservado 5 = SBus
		Len	Número de itens dos dados de saída de processo
		PO1	Dados de saída de processo 1
		PO2	Dados de saída de processo 2
		PO3	Dados de saída de processo 3
		PO4	Dados de saída de processo 4
		PO5	Dados de saída de processo 5
		PO6	Dados de saída de processo 6
		PO7	Dados de saída de processo 7
		PO8	Dados de saída de processo 8
		PO9	Dados de saída de processo 9
		PO10	Dados de saída de processo 10
_MovCommDef (somente com MQx)	MOVCOM	Bus types	**Tipo da rede (interface)** ML_BT_S1: S1 (RS-485 #2"
		address	**Endereço individual ou endereço de grupo para o MOVIMOT® ser endereçado**
			0 ... 99 Endereçamento simples
			100 ... 199 Endereçamento de grupo
			255 Transmissão
		Format	**Entrada de dados de processo para transferência de dados** 3 = 2 palavras de dados de processo acíclicas (para MOVIMOT®) = ML_FT_2 5 = 3 palavras de dados de processo acíclicas (para MOVIMOT®) = ML_FT_3
		PdPointer	**Número da variável H' a qual os dados de processo são armazenados ou da qual os dados a ser escritos são obtidos**. (A estrutura de dados para H' é descrita abaixo com detalhes.)
		ParaPointer	**Número da variável H' a qual os dados de parâmetro são armazenados ou da qual os dados a ser escritos são obtidos**. O MOVIMOT® não realiza esta função.

Comando	Estrutura padrão	Elementos	Descrição breve
_MoviLink	MOVLNK	Bus types	Possíveis tipos de rede: ML_BT_S0: S0 (RS485 #1) ML_BT_S1: S1 (RS485 #2) ML_BT_SBUS: SBus
		Address	Endereço simples (0...99) ou endereço de grupo (100...199)
		Format	Especificação do processo (PD) e canais de parâmetro (PARAM) para transferência de dados: ML_FT_PAR1: PARAM+1PD ML_FT_1: 1PD ML_FT_PAR2: PARAM+2PD ML_FT_2: 2PD ML_FT_PAR3: PARAM+3PD ML_FT_3: 3PD ML_FT_PAR: Parameter (sem PD)
		Service	Serviço de comunicação ML_S_RD: Serviço de leitura ML_S_WR: Escrita com armazenagem na memória não-volátil ML_S_WRV: Escrita sem salvar
		Index	Número Index do parâmetro a ser modificado ou lido (ver diretório index de parâmetro)
		DPointer	Número da variável a qual os dados de leitura são armazenados ou a qual os dados a ser escritos são obtidos (estrutura MLDATA)
		Result	Contém o código de erro após o serviço de parâmetro ter sido realizado, ou contém zero se não houver erro
	MLDATA	WritePar	Parâmetro que é enviado para serviços de escrita
		ReadPar	Parâmetro que é enviado para serviços de leitura
		PO1	Dados de saída de processo 1
		PO2	Dados de saída de processo 2
		PO3	Dados de saída de processo 3
		PI1	Dados de entrada de processo 1
		PI2	Dados de entrada de processo 2
		PI3	Dados de entrada de processo 3
_SBusCommDef	SCREC	ObjectNo	Número do objeto
		Format	Número dos bytes de dados e do formato de dados
		DPointer	Número da variável de que ponto os dados recebidos são armazenados
	SCTRACYCL	ObjectNo	Número do objeto
		Format	Número dos bytes de dados e do formato de dados
		DPointer	Número da variável de onde começa os dados a ser enviados
	SCTRCYCL	ObjectNo	Número do objeto
		CycleTime	Duração do ciclo [ms] Duração do ciclo válida: - 1, 2 ... 9 ms - 10, 20, ... 65530 ms
		Offset	Duração do Offset [ms] Duração do offset válida: - 0, 1, 2 ... 65534 ms para duração do ciclo < 10 ms - 0, 10, 20, ... 65530 ms para duração do ciclo >= 10 ms
		Format	Número dos bytes de dados e do formato de dados
		DPointer	Número da variável de onde começa os dados a ser enviados
		Result	Código de retorno para parâmetros de ajuste
_SetSys	SSPOSRAMP	Up	Rampa de aceleração (ms)
		Down	Rampa de desaceleração (ms)

Comando	Estrutura padrão	Elementos	Descrição breve
	SSPOSSPEED	CW	Rotação de posicionamento horária (0.1 U/min)
		CCW	Rotação de posicionamento antihorária (0.1 U/min)
	SSPIDATA3	Len	Número de itens dos dados de entrada de processo a ser enviados
		PI1	Dados de entrada de processo 1
		PI2	Dados de entrada de processo 2
		PI3	Dados de entrada de processo 3
	SSPIDATA10	Len	Número de itens dos dados de entrada de processo a ser enviados
		PI1	Dados de entrada de processo 1
		PI2	Dados de entrada de processo 2
		PI3	Dados de entrada de processo 3
		PI4	Dados de entrada de processo 4
		PI5	Dados de entrada de processo 5
		PI6	Dados de entrada de processo 6
		PI7	Dados de entrada de processo 7
		PI8	Dados de entrada de processo 8
		PI9	Dados de entrada de processo 9
		PI10	Dados de entrada de processo 10

Estas estruturas padrão são utilizadas conforme a seguir. Primeiro, a variável é declarada como a variável da estrutura na parte de declaração. Então os elementos da estrutura são endereçados como explicado no exemplo a seguir. A estrutura é endereçada dentro do comando utilizando o nome da variável da estrutura sem acréscimos.

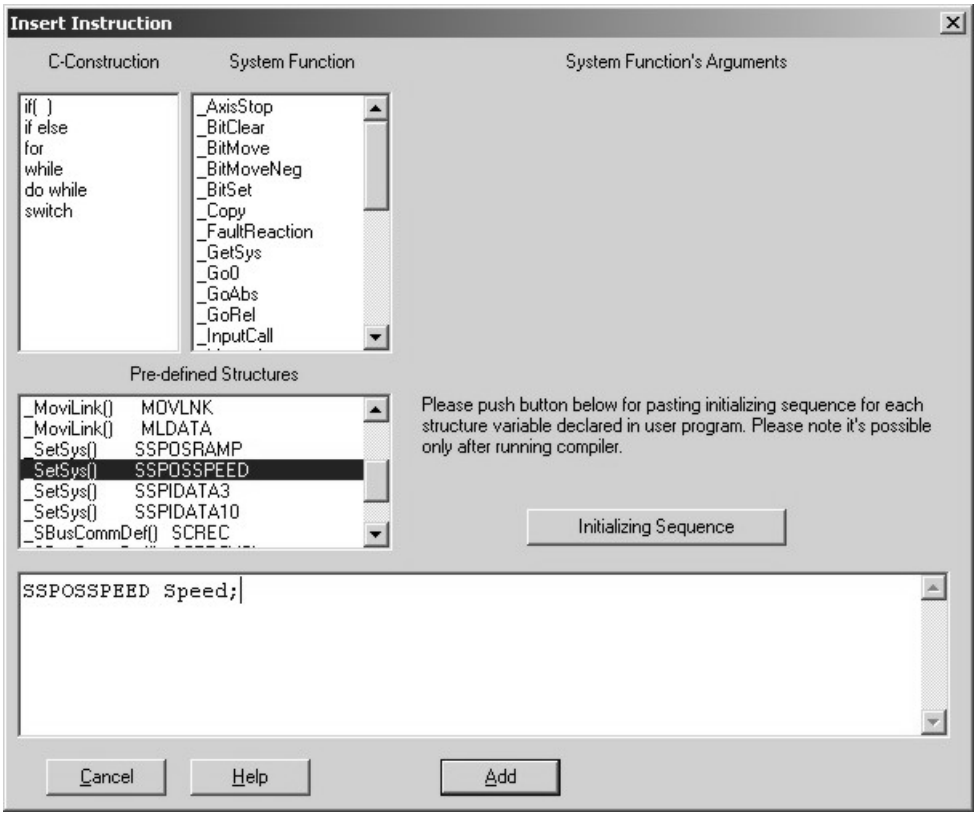
Exemplo: As rotações de posicionamento são ajustadas utilizando o comando _SetSys.

```
// Declare
SSPOSSPEED rapid speed, slow speed;

// Initiate
rapid speed.cw = 14000;     // rapid speed cw 1400 rpm
rapid speed.ccw = 12500;    // rapid speed ccw 1250 rpm
slow speed.cw = 3000;       // slow speed cw 300 rpm
slow speed.ccw = 4500; // slow speed ccw 450 rpm

// set rapid speed
_SetSys( SS_POSSPEED,rapid speed );

// set slow speed
_SetSys( SS_POSSPEED,slow speed );
```

A ferramenta de inserção pode ser utilizada para apoio quando divulgar e inicializar estruturas padrão. O cursor deve ser posicionado na janela Editor no lugar onde a declaração para as variáveis da estrutura deve ser inserida. Acessar a ferramenta clicando com o botão direito do mouse para abrir o conteúdo do menu e então selecionar [Insert Instruction].

Pode-se também acessar a ferramenta de inserção pressionando o ícone na barra de ferramentas ou escolhendo [Edit] / [Insert Instruction] da barra de menu. Selecionar a estrutura pré-definida. O nome da variável agora pode ser modificado ou editado na janela de edição. Se forem declaradas várias variáveis da estrutura a partir do mesmo tipo de estrutura, separá-las com uma vírgula.

Depois de todas as variáveis da estrutura terem sido declaradas elas devem ser inicializadas dependendo da aplicação específica. A ferramenta de inserção também pode ser utilizada para este processo. Colocar o cursor na janela Editor no lugar onde a seqüência de inicialização deve ser acrescentada. Compilar o programa e então acessar a ferramenta de inserção. Pressionar a tecla [Initialising Sequence]. É criado um bloco de inicialização para cada variável da estrutura que foi declarada.



06096AXX

Figura 60: Inserindo um comando

11.9 Estruturas definidas pelo usuário

Os usuários podem definir suas próprias estruturas além das estruturas padrão SEW. Primeiro, a estrutura deve ser criada. Isto é feito na parte de declaração do programa. É utilizada a palavra chave 'typedef struct'. Isto pode ser explicado levando-se em consideração o exemplo que cria uma tabela de posição.

```
// Define user structure
typedef struct
{
  long pos1;
  long pos2;
  long pos3;
  long pos4;
  long pos5;
} table;
```

Este cria uma estrutura com o nome `table`. Você pode agora utilizar esta estrutura como explicado para as estruturas padrão. A próxima fase é declarar uma variável como variável da estrutura.

```
// Declare structure
table postable;
```

Agora a variável PosTable foi declarada como uma variável da estrutura do tipo de estrutura `table`. A próxima fase é acessar os elementos. Para isto, a tabela deve ser inicializada.

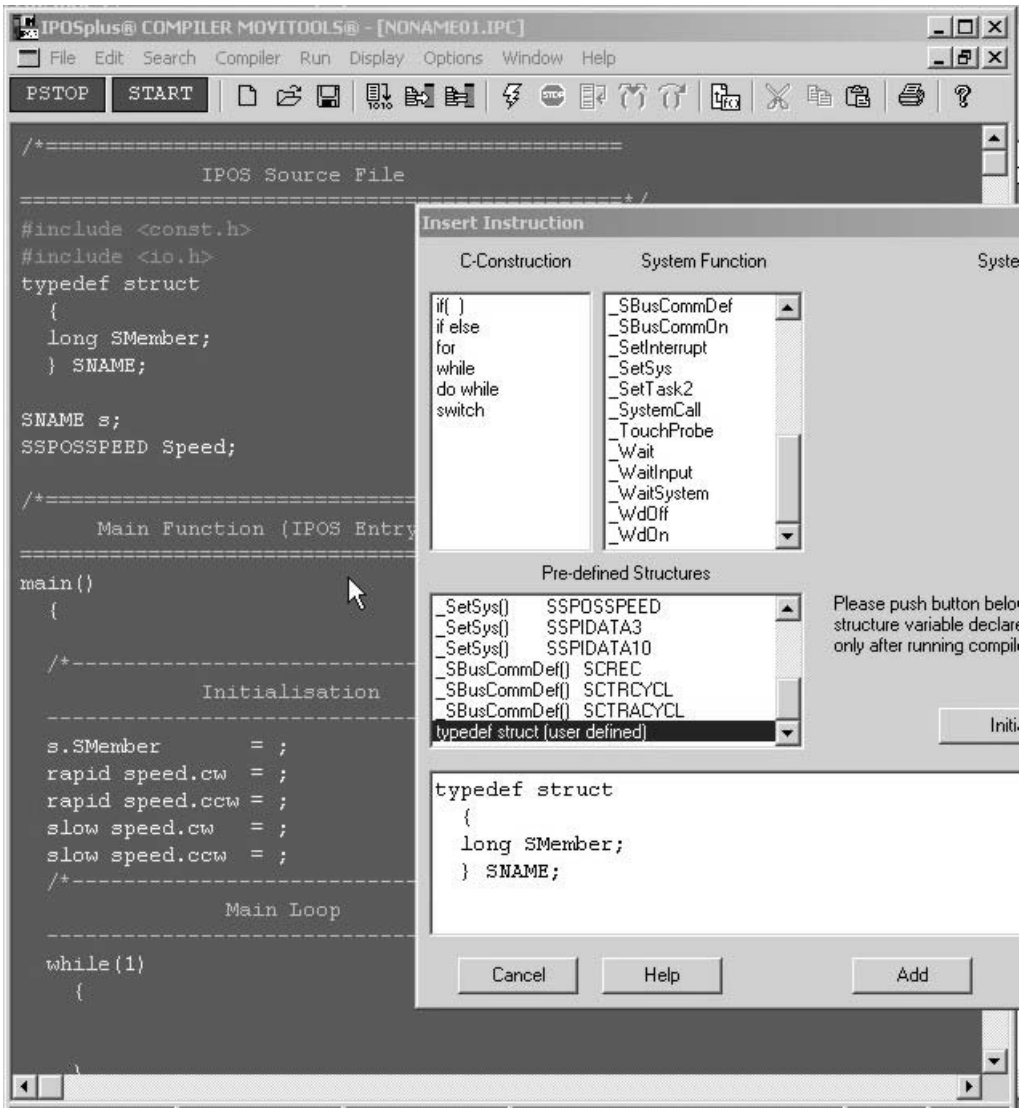
```
// Initiate
postable.pos1 = 100000;
postable.pos2 = 120000;
postable.pos3 = 50000;
postable.pos4 = 200000;
postable.pos5 = 10000;
```

A seguir, uma descrição geral do procedimento para a elaboração de uma estrutura do usuário:

```
typedef struct
{
  Type Identifier1;
  Type Identifier2;
  ...
  Type IdentifierN;
} StructureName;

StructureName VariableName;

VariableName.Identifier1 = ...;
VariableName.Identifier2 = ...;

VariableName.IdentifierN = ...;
```

A tabela de inserção também pode ser utilizada para estruturas definidas pelo usuário. Para isto, escolher typedef struct (definido pelo usuário) da janela de estruturas predefinidas da ferramenta de inserção. Agora você pode mudar o nome dos elementos e da estrutura.

Depois de ter inserido a estrutura, é acrescentada uma linha de declaração na janela Editor que declara as variáveis deste tipo de estrutura. Esta linha deve ainda ser editada no Editor. Os elementos adicionais podem ser acrescentados do mesmo modo, dentro do tipo de estrutura. Uma vez as entradas terem sido compiladas, pode ser ativada uma seqüência de inicialização utilizando a ferramenta de inserção. Antes disso, o cursor deve estar posicionado em Editor no lugar onde é para ser inserida a seqüência de inicialização.



06097AXX

Figura 61: Estruturas

11.10 long

Como alternativa para atribuição das variáveis utilizando #define, a palavra-chave long pode ser utilizada para declarar uma variável individual. Neste caso, o número da variável é atribuído pelo compilador durante a compilação.

A palavra-chave long inicia a declaração de uma ou mais variáveis globais. O exemplo a seguir mostra como utilizar a palavra-chave.

A sintaxe da declaração de uma ou mais variáveis globais é a seguinte:

```
long Identifier1 [, Identifier n] ;
```

Exemplo:

```
long setpoint,actual_value;
```

Durante a compilação, o ajuste das variáveis simbólicas e o valor atual são atribuídos a uma variável IPOS$^{plus®}$. O usuário sempre acessa a variável utilizando o nome simbólico.

11.11 initial long

'initial long' é disponível como outra palavra-chave. 'initial long' declara uma variável que é armazenada na faixa de H0 até H127 durante a compilação. Isto significa que a variável é armazenada na faixa de variáveis que não é perdida quando há uma falha de potência.

Exemplo:

```
initial long start position, end position;
```

A legibilidade do texto do programa é melhorada significativamente se todas as constantes forem escritas em letra maiúscula (por ex. SECOND, MAXIMUM, etc.) e as variáveis forem escritas em maiúscula/minúscula (por ex. SpeedSetpoint, PositionCW, etc.).

11.12 #pragma

A diretiva #pragma pode ser utilizada para influenciar a faixa variável ocupada pelas palavras-chave 'long' e 'initial long'.

```
Syntax: #pragma  Directive  Parameter1 Parameter2 ...
```

O compilador realiza as seguintes diretivas pragma:

#pragma list	Faz com que as linhas do texto fonte sejam incluídas como comentários no programa IPOS$^{plus®}$ resultante.
#pragma var Hmin Hmax	Instrui o compilador a utilizar variáveis IPOS$^{plus®}$ Hmin por Hmax como variáveis auxiliares para expressões de cálculo. Hmax deve ser maior do que Hmin. Se o programador utiliza as mesmas variáveis no programa, é produzida uma mensagem de erro. A experiência mostra que o compilador necessita cerca de 10 variáveis auxiliares. O compilador utiliza variáveis H400 por H419 se esta diretiva não for especificada explicitamente.
#pragma globals Hmin Hmax	Instrui o compilador a atribuir um número de variável da faixa de variável Hmin até Hmax para as variáveis declaradas com a palavra-chave long. O usuário é responsável por evitar sobreposições quando ligar os nomes variáveis com símbolos utilizando #define. O compilador utiliza variáveis H420 até H449 se esta diretiva não for especificada explicitamente.
#pragma initials Hmin Hmax	Instrui o compilador a atribuir os números Hmin até Hmax para as variáveis globais declaradas com a palavra-chave inicial. As variáveis iniciais são variáveis H0 até H127 que são armazenadas quando a alimentação é desligada. Exceção: H0 até H15 para MDS, MDV, MCS, MCV com cam disk. O compilador utiliza H0 até H127 se esta diretiva não for especificada explicitamente.

Já que a faixa da variável H360 até H450 para a opção versão aplicação "Synchronous operation" e "Cam Disk" é atribuída a variáveis adicionais do sistema, recomendamos sempre atribuir as variáveis auxiliares e globais com a diretiva #pragma em uma faixa diferente.

Exemplo:

```
#pragma var 350,365
#pragma globals 130,160
#pragma initials 10 30

long pos speed cw, pos speed ccw;
initial long start position, end position;
```

Estas linhas fazem com que o compilador utilize as variáveis IPOS$^{plus®}$ de H350 em diante como variáveis auxiliares. As variáveis PosSpeedCW e PosSpeedCCW que são declaradas com a palavra-chave 'long' agora são armazenadas nas variáveis IPOS$^{plus®}$ entre H130 e H160 por causa das linhas de comando globais 130 160 #pragma. As variáveis StartPosition e EndPosition que são declaradas com a palavra-chave 'initial long' agora são armazenadas nas variáveis IPOS$^{plus®}$ entre H10 e H30 por causa das linhas de comando iniciais 10 30 #pragma. Desde que elas estejam na faixa variável de H0 até H127, estas variáveis também podem ser armazenadas na memória não-volátil.

Os **números variáveis** são utilizados **sem** a letra anterior **H** quando eles são utilizados dentro das diretivas #pragma.

11.13 Explicação de const.h and io.h / constb.h and iob.h

O arquivo de cabeçalho const.h define muitos identificadores úteis. Somente os nomes simbólicos das variáveis do sistema são inicialmente importantes. Os outros identificadores são importantes para usuários hábeis que não utilizam mais a ferramenta de inserção. Esta seção contém as definições de argumentos para acesso às funções padrão.

O arquivo de cabeçalho io.h e o arquivo const.h são arquivos pré-definidos que não podem ser mudados. O arquivo é impresso abaixo:

```
/*============================================
File name: Io.h
File version: 2.01

Definition of bitmasks for digital in- and outputs

Please do not modify this file!

(C) 1999 SEW-EURODRIVE
============================================*/

#ifndef _IO_H
#define _IO_H

#define DI00 (H483 & 0b1)
#define DI01 (H483 & 0b10)
#define DI02 (H483 & 0b100)
#define DI03 (H483 & 0b1000)
#define DI04 (H483 & 0b10000)
#define DI05 (H483 & 0b100000)
#define DI10 (H483 & 0b1000000)
#define DI11 (H483 & 0b10000000)
#define DI12 (H483 & 0b100000000)
#define DI13 (H483 & 0b1000000000)
#define DI14 (H483 & 0b10000000000)
#define DI15 (H483 & 0b100000000000)
#define DI16 (H483 & 0b1000000000000)
#define DI17 (H483 & 0b10000000000000)

#define DB00 (H482 & 0b1)
#define DO01 (H482 & 0b10)
#define DO02 (H482 & 0b100)
#define DO10 (H482 & 0b1000)
#define DO11 (H482 & 0b10000)
#define DO12 (H482 & 0b100000)
#define DO13 (H482 & 0b1000000)
#define DO14 (H482 & 0b10000000)
#define DO15 (H482 & 0b100000000)
#define DO16 (H482 & 0b1000000000)
#define DO17 (H482 & 0b10000000000)

#endif
```

O arquivo de cabeçalho io.h define macros que facilitam verificar os níveis dos bornes. O exemplo a seguir ilustra esta função.

```
if( DI00 )
{
  H1 = 1;  // execute command block, if terminal DI00 has NOT level 0
}
else
{
  H1 = 0;  // execute command block, if terminal DI00 has level 0
}
```

O comando *if* verifica o borne DI00, borne /CONTROLLER INHIBIT. Se o argumento do comando *if* for zero, então os comandos na outra parte são processados (supõe que há uma outra parte). Neste caso, a variável IPOS$^{plus®}$ H1 é ajustada para zero ou um, dependendo do nível de entrada do borne DI00. Observe que é impossível verificar quando o borne é ajustado para 1 (DI00 == 1) porque a macro fornece uma evolução digital. Em termos práticos, é possível verificar se a entrada está em nível zero (DI00 == 0) ou diferente de zero (DI00 != 0).

Este extrato do programa pode ser feito utilizando os comandos que já foram explicados. Isto é feito introduzindo identificadores simbólicos adicionais.

```
#define controller inhibit    H1
#define HI                    1
#define LO 0

if( DI00 )
{
  controller inhibit = 1;  // execute command block, if terminal DI00 has NOT level 0
}
else
{
  controller inhibit = 0;  // execute command block, if terminal DI00 has level 0
}
```

Aparece o seguinte na janela de variáveis:

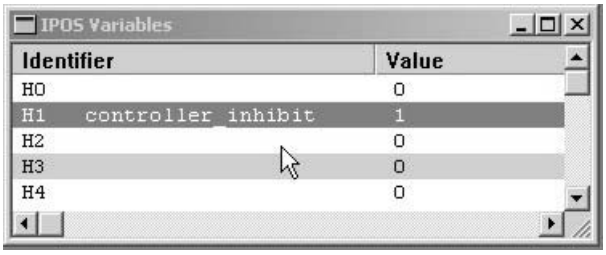


Figura 62: Janela de variáveis

Observe que o arquivo de cabeçalho io.h deve ser vinculado utilizando a linha do comando #include io.h antes de ser utilizado.

11.14 Identificadores

Embora já tenhamos utilizado os identificadores várias vezes, esta seção fornece informação adicional sobre eles. Um identificador é entendido como o nome que pode ser adotado por um identificador Macro (seção Define), um nome simbólico da variável ou um nome da função. Pode ser utilizado no identificador somente letras, números e _, e o identificador deve iniciar com uma letra ou um _. O identificador pode ter até 32 caracteres no comprimento.

São válidos os seguintes identificadores:	Os nomes a seguir não são identificadores:
Terminal X13_4 Setpoint1 _Control_word	TerminalX13.4 Setpoint 1 Control word 1st setpoint 1_Input My function ThisIdentifierNameIsMuchTooLong

O compilador IPOS$^{plus®}$ é case-sensitive.

11.15 Constantes

O compilador IPOS$^{plus®}$ realiza vários tipos de constantes que são diferenciadas no texto fonte por suas notações específicas. A representação em formatos diferentes pode melhorar a legibilidade do texto fonte dependendo de como eles são utilizados.

Os formatos decimal, hexadecimal ou binário são formas possíveis de representação.

As constantes hexadecimais iniciam com o caracter '0x', as constantes binárias com o caracter '0b'. Aqui estão alguns exemplos:

Constantes decimais	Constantes hexadecimais		Constantes binárias	
123	0x23	= 35 dec	0b000100	= 4 dec
–50	0xabc	= 2748 dec	0b10	= 2 dec
030	0xFFFFFFFF	= –1 dec	0b11111111	= 255 dec

11.16 Variáveis IPOS$^{plus®}$ no compilador

As variáveis IPOS$^{plus®}$ são praticamente um elemento de linguagem e não podem ser declaradas explicitamente. Todas elas têm o mesmo tipo de dados (32 bit com sinal) e são válidas globalmente em todo o texto fonte. A linha a seguir está presente implicitamente em cada módulo:

```
long H0, H1, H2, H3, ... , H1023.
```

Para identificar as variáveis simbolicamente, as diretivas #define ou #declare podem ser utilizadas para definir um nome simbólico.

Exemplo

```
#define TESTVAR1 H73 //H73 é dado o nome simbólico "TESTVAR1"
```

H73 recebe o valor 134 no programa de uma de 3 programações:

```
TESTVAR1 = 134;
TESTVAR1 = 0x86;
TESTVAR1 = 0b10000110;
```

11.17 Declaração das variáveis globais

Outra opção é declarar as variáveis globais com a palavra-chave long como já explicado acima. O compilador então define automaticamente os números das variáveis (ver #pragma). Os números das variáveis são atribuídos em ordem crescente de acordo com onde as divulgações das variáveis ocorrem no texto fonte. A divulgação inicia com a palavra-chave long, seguida pela lista dos identificadores simbólicos separada por vírgulas. A declaração termina com um ponto-e-vírgula. A declaração pode se extender sobre várias linhas do texto fonte.

A variável global pode ser declarada em qualquer lugar no programa, desde que esteja fora dos blocos (geralmente funções). Por razões de clareza, as variáveis devem ser declaradas no início do módulo de texto fonte. Uma variável global também deve ser declarada antes de ser utilizada.

Exemplos:

```
long a, b;                  // Variables are available in the range
long this_is_a_variable;    // defined for global variables
long c, d;
```

O compilador atribui as variáveis H420 até H424 aos identificadores a, b, this_is_a_variable, c e d. No exemplo a seguir, a diretiva '#pragma globals' instrui o compilador a atribuir as variáveis H150 até H160 aos identificadores.

```
#pragma globals 150,160
long a, b;
long this_is_a_variable;
long c, d;
```

Não são **permitidas** declarações múltiplas das variáveis globais:

```
long a, b, c;
long d, a;
```

É uma boa idéia utilizar a opção de definição das variáveis globais quando o número da variável não é importante para a aplicação. Geralmente isto acontece porque as variáveis são constantemente acessadas pelos símbolos. Além disso, a declaração da variável global sublinha a modularidade e facilita reutilizar módulos. As estruturas padrão ou definidas pelo usuário podem ser utilizadas como um recurso se forem necessários grupos de variáveis, por ex. para funções do sistema.

Para uma descrição das variáveis do sistema H473 até H511, consultar a seção "Resumo das Variáveis do Sistema." É dada no apêndice uma lista de variáveis do sistema e seus identificadores simbólicos.

11.18 Endereçamento indireto – indicador

As denominações *H0 até *H511 também são permitidas como nomes das variáveis para empregar o IPOS$^{plus®}$ e acessar indiretamente as variáveis (SET [H] = H). Estes nomes podem ser utilizados nos lados direito e esquerdo das programações ou nas expressões, como H0 até H511. Neste caso, no entanto, o compilador insere os comandos indiretos.

Exemplo para utilizar variáveis indiretas:

```
H2 = 5;
H3 = 6;

H5 = 7;
H6 = 3;

H1 = *H2 + *H3;
```

Variável H1 é atribuído o valor 10 porque o valor 7 é acessado indiretamente via *H2 (ver H5) e o valor 3 é acessado indiretamente via *H3 (ver H6).

11.19 numof()

A palavra-chave numof() volta o número de uma variável. O identificador de uma variável direta ou simbólica é dado como argumento. Não é permitido o argumento ser uma expressão composta.

```
#define setpoint H200
#declare setpoint2 setpoint:1

H1 = numof(H7);
H2 = numof (setpoint);
H3 = numof (setpoint2);
```

Estas linhas do programa alimentam as variáveis IPOS$^{plus®}$ com os seguintes valores:

```
H1 = 7
H2 = 200
H3 = 201
```

Não são **permitidas** as seguintes linhas do programa:

```
#define Setpoint H10+H30
#define Var1 H200

H1 = numof(*H1);
H1 = numof(H1 + H4);
H1 = numof(H3 + 6);
H1 = numof(Setpoint);
H1 = numof(Var1:1);
```

12 Compiler – Operadores

Os operadores são utilizados para ligar um identificador a outro e comandar, a fim de realizar certas operações. O compilador IPOS$^{plus®}$ fornece operadores para a realização de operações aritméticas, operações de bit, operações de programação ou de comparação.

Os operadores são divididos em várias categorias e possuem uma ordem específica de prioridade. A ordem de prioridade determina a ordem a qual os operadores são realizados dentro de um comando. A tabela a seguir indica todos os operadores que podem ser realizados pelo compilador IPOS$^{plus®}$ por ordem de prioridade.

12.1 Ordem de prioridade dos operadores

Categoria	Operador	Descrição
1.	()	Brackets
2. Unary	! ~ + – ++ --	Logical negation (NOT) bit-by-bit complement Unary plus Unary minus Pre or post-incrementing Pre or post-decrementing
3. Multiplicative	* / %	Multiplication Integer division Modulo remainder
4. Additive	+ -	Binary plus Binary minus
5. Shift	<< >>	Shift left Shift right
6. Relational	< <= > >=	Less than Less than or equal to Greater than Greater than or equal to
7. Equality	== !=	Equal to Not equal to
8.	&	Bit-by-bit AND
9.	^	Bit-by-bit XOR
10.	|	Bit-by-bit OR
11.	&&	Logical AND
12.	||	Logical OR
13. Conditional	? :	Ternary operators, see section "Ternary Operators"
14. Assignment	= *= /= %= += –= &= ^= |= <<= >>=	Simple assignment Assign product Assign quotient Assign remainder Assign sum Assign difference Assign bit-by-bit AND Assign bit-by-bit XOR Assign bit-by-bit OR Assign shift left Assign shift right
15. Comma	,	Evaluate

A categoria 1 tem prioridade mais alta, a categoria 2 (operadores unary) tem a segunda prioridade mais alta, etc. O operador *comma* tem prioridade mais baixa.

Os operadores na mesma categoria têm a mesma posição.

Os operadores unary (categoria 2), operadores conditional (categoria 13) e os operadores assignment (categoria 14) atribuem da direita para a esquerda; todos os outros atribuem da esquerda para a direita.

O operador para multiplicação (*) é posicionado antes do operador para adição (+), assim a multiplicação é realizada antes da adição no seguinte comando:

```
H1 = 3 * 7 + 2 * 4;
```

H1 recebe o valor 29.

Devem ser utilizados ( ) se a adição for realizada antes da multiplicação:

```
H1 = 3 * ( 7 + 2 ) * 4;
```

H1 recebe o valor 108 (a partir de 3 * 9 * 4).

A seqüência pode ser forçada pelos ( ) para assegurar que a seqüência de operação seja a desejada. São permitidos ( ) aninhados. Os ( ) desnecessários não têm efeito na função do programa.

12.2 Operadores Unary

Os operadores unary são posicionados antes ou após um operando e influenciam somente este operando.

```
H1 = -H2; // The unary minus operator (-) forms the value of H2 with a changed sign
H1 = ~H2; // The unary complement operator (~) forms the bit-by-bit complement of H2
++H3 //, the pre-increment operator (++) increases the value of H3 by one
```

12.3 Operadores Binary

Estes operadores ligam dois operandos juntos e são localizados entre dois operandos.

Exemplo

```
H1 = H2;      // The binary assignment operator (=) assigns variable H1 the value of H2
H1 = H2 - 3;  // The binary minus operator (-) forms the difference between H2 and 3
```

Os operadores assignment combinados conduzem a uma notação abreviada. Embora eles tornam mais difícil ler um programa, eles são mencionados por causa da integralidade. A operação é realizada no exemplo com H1 = 2 (0b10) e H2 =3 (0b11).

Operador	Operação	Exemplo	Corresponde a	Exemplo H1 =
=	Programação simples	H1 = H2;	H1 = H2;	3
*=	Atribui o produto	H1 *= H2;	H1 = H1 * H2;	6
/=	Atribui o quociente	H1 /= H2;	H1 = H1 / H2;	0
%=	Atribui o restante	H1 %= H2:	H1 = H1 % H2;	2
+=	Atribui a soma	H1 += H2;	H1 = H1 + H2;	5
–=	Atribui a diferença	H1 –= H2;	H1 = H1 – H2;	–1
&=	Atribui bit-a-bit AND	H1 &= H2;	H1 = H1 & H2;	0b10
^=	Atribui bit-a-bit XOR	H1 ^= H2;	H1 = H1 ^ H2;	0b01
|=	Atribui bit-a-bit OR	H1 |= H2;	H1 = H1 | H2;	0b11
<<=	Atribui o deslocamento à esquerda	H1 <<= H2;	H1 = H1 << H2;	0b1000
>>=	Atribui o deslocamento aritmético à direita	H1 >>= H2;	H1 = H1 >> H2;	0b0

12.4 Operadores Ternary

O compilador IPOS$^{plus®}$ somente reconhece um operador que liga três operandos juntos: o operador conditional. Sua forma é a seguinte:

a ? b : c significa: Se a é verdadeiro então b, senão c

onde a é uma expressão lógica e b e c são expressões.

Exemplo

```
H1 = H2 == 3 ? H3 : H4;    // If H2 equals 3, H1 is assigned the value of H3,
                           // otherwise it is assigned the value H4
```

O exemplo é a notação abreviada de:

```
if (H2 == 3)
  H1 = H3;
else
  H1 = H4;
```

Onde possível, o operador ternary não deve ser utilizado devido a ilegibilidade do código fonte.

13 Compiler – Construções

O compilador IPOS$^{plus®}$ fornece construções que também são disponíveis em outras linguagens de alto nível.

São disponíveis as seguintes construções:

- if...else
- for
- while
- do...while
- switch...case...default

Estas são completadas por comandos como ’continue’ e ’break’, que são utilizados como elementos de controle dentro destas construções.

13.1 if...else

Sintaxe

```
if ( Expression )
// Statement
else
// Statement
```

As palavras-chave if e else controlam o fluxo do programa dependendo se a expressão seguinte a palavra-chave if volta o valor TRUE (não igual a zero) ou FALSE (igual a zero). A derivação else é opcional. Isto é realizado se a expressão volta o valor FALSE. Em caso especial, o comando também pode ser um bloco o qual vários comandos podem ser especificados. Neste caso, o comando bloco deve ser incluso pelos sinais { } ({ statement block})..

Sem derivação else	Com derivação else	Com bloco como derivação if	Com bloco como derivação else

`if ( H1 == 3 )`
`  H2 = 10;`
	`if ( H1 == 3 )`
	`  H2 = 10;`
	`else`
	`  H2 = 8;`
		`if ( H1 > 3 )`
		`{`
		`  H2 = 10;`
		`  H3 = 11;`
		`}`
			`if ( H1 > 3 )`
			`  H2 = 9;`
			`else`
			`{`
			`  H2 = 10;`
			`  H3 = 11;`
			`}`

A expressão também pode ser composta de várias condições interligadas logicamente. Conseqüentemente, a lógica AND (&&) e a lógica OR (||) são disponíveis como operações lógicas.

Atenção: A ; (ponto e vírgula) no final de um comando *if* sempre torna a condição verdadeira.

Exemplo

```
if ( ( H1 >= 3 ) && ( H1 <= 12 ) )
  H2 = 10;
```

A variável H2 é ajustada para o valor 10 se H1 for maior ou igual a 3 e também for menor ou igual a 12. Em outras palavras: H2 é ajustada para o valor 10 se o valor da H1 estiver entre 3 e 12.

Os sinais internos não são necessários, mas eles aumentam a legibilidade do programa.

Exemplo

```
if ( H1 < 2 || H1 > 14 )
H  2 = 10;
```

A variável H2 é ajustada para o valor 10 se H1 for menor do que 2 ou maior do que 14. Em outras palavras: H2 é ajustada para o valor 10 se o valor da H1 não estiver entre 2 e 14.

13.2 for

Sintaxe

```
for ( Expression1 ; Expression2 ; Expression3 )
// Statement
```

O comando *for* pode ser utilizado para construir rotinas do programa que devem ser canceladas após um número específico de repetições. A seguir, o significado das três expressões:

Expression1 é realizada uma vez no início da rotina *for*, onde o funcionamento das variáveis é inicializado. Expression2 determina quando a rotina é interrompida. A rotina é interrompida se a expressão volta o valor lógico FALSE (ou igual a zero). Expression3 é processada após o comando ter sido realizado. Geralmente, é utilizada para alterar a variável run. O comando forma o corpo da rotina que pode consistir de um comando ou um bloco de comando.

Exemplo

```
H1 = 20;
for ( H0 = 0; H0 < 10; ++H0 )
  H1 = H1 + 2;
```

H0 é ajustada para zero no início. Então é realizada uma verificação para ver se H0 alcançou o valor 10. Se este não for o caso, então o comando é processado. Neste exemplo, portanto, H1 é aumentada por 2. A variável run H0 é então aumentada por um. A seguir, a verificação se H0 alcançou o valor 10 é repetida, e assim por diante.

No final da rotina, o valor da H0 é 10 e da H1 é 40 porque a rotina é realizada 10 vezes (contador da rotina H0 funciona de 0 a 9 e depois é cancelado).

Exemplo de um bloco de comando

```
H1 = 20;
H2 = 0;
for ( H0 = 0; H0 < 10; ++H0 )
{
  H1 = H1 + 2;
  ++H2;
}
```

Ao passo que a variável H1 é aumentada por 2 cada vez que a rotina funciona completamente, a variável H2 somente é aumentada por 1 (++H2 significa pré-incremento).

Se o comando *continue* é processado dentro do bloco de comando, isto quer dizer que o programa pula para o final do bloco de comando e então processa a expression3, que ativa a verificação para determinar se a condição no comando *for* é realizada para um novo ciclo da rotina.

Exemplo

```
H1 = 20;
H2 = 0;
for ( H0 = 0; H0 < 10; ++H0 )
{
  H1 = H1 + 2;
  if ( H1 > 32 )
    continue;
  ++H2;
}
```

A verificação *if* com o comando *continue* quer dizer que a variável H2 não é mais incrementada, assim que a H1 for maior do que 32. Isto quer dizer que quando a rotina é terminada, o valor da variável H1 é 40 e da H2 é 6.

Se o comando *break* é processado no bloco de comando, isto significa que o programa sai da rotina *for* naquele ponto. A rotina não é mais continuada.

Exemplo

```
H1 = 20;
H2 = 0;
for ( H0 = 0; H0 < 10; ++H0 )
{
  H1 = H1 + 2;
  if ( H1 > 32 )
    break;
  ++H2;
}
```

A verificação *if* com o comando *break* quer dizer que a rotina é cancelada assim que a H1 for maior do que 32. Isto quer dizer que quando a rotina é cancelada, o valor da variável H1 é 34 e da H2 é 6.

13.3 while

Sintaxe

```
:  while ( Expression )
// Statement
```

O comando *while* é uma rotina condicional que realiza o comando *for*, contanto que o valor da expressão seja TRUE (não igual a zero). O comando nunca é realizado se a expressão não tiver o valor TRUE. A expressão sempre é processada antes do comando.

O comando também pode ser um bloco de comando o qual vários comandos podem ser especificados.

A expressão também pode ser composta de várias condições interligadas logicamente.

Exemplo

```
H2 = 0;
H1 = 10;
while ( H1 > 5 )
{
  H2 = H2 + 1;
  --H1;
}
```

Os comandos dentro do bloco são cumpridos contanto que a H1 seja maior do que 5. A H2 recebe valor 5 quando a rotina pára.

Como na rotina *for*, aqui também é possível utilizar os comandos *break* e *continue*. O comando *break* mais uma vez faz com que a rotina *while* seja cancelada. O comando *continue* resulta em um salto para o final do bloco de comando seguido pela repetição da verificação de expressão, para ver se a rotina será processada novamente.

Exemplo

```
H1 = 0;
while ( H1 < 20 )
{
  ++H1;
  if ( H1 > 10 )
    continue;
  H2 = H2 +2;
}
```

Contanto que a H1 seja menor ou igual a 10, a H2 é aumentada por 2. Se o valor da H1 é maior do que 10, o processamento salta para o final da rotina *while*, no ponto que a condição para funcionar através da rotina é novamente testado. H2 não é mais mudada durante a adição dos ciclos de rotina. Se o valor da H1 é 20, a rotina é cancelada.

Exemplo

```
H1 = 0;
while(1)
{
  ++H1;
  if ( H1 == 20 )
    break;
  if ( H1 > 10 )
    continue;
  H2 = H2 +2;
}
```

Este exemplo tem o mesmo efeito daquele mostrado anteriormente. A "endless loop" é cancelada utilizando o comando *break* se a H1 for igual a 20.

Esta é uma rotina contínua, que pode ser criada utilizando a seguinte construção:

```
while(1)
// Statement
```

Isto é porque a expression 1 volta sempre o valor TRUE.

O Task1 roda tipicamente em uma rotina contínua deste tipo.

13.4 do...while

Sintaxe

```
do
// Loop statements
while ( Expression );
```

O comando *do* é uma rotina condicional a qual a condição *break* é verificada no final da rotina. Como resultado, esta rotina com o comando *do*, sempre realiza pelo menos uma iteração (pelo menos uma volta completa).

Primeiro, o comando é realizado. Como um bloco de comando, ele também pode conter vários comandos. Isto é seguido pelo teste se a expressão tem o valor TRUE (diferente de zero) ou FALSE (igual a zero). Se o valor é TRUE, o comando é novamente realizado, caso contrário a rotina é cancelada.

A expressão também pode ser composta de várias condições interligadas logicamente.

Ao contrário da rotina *while*, o comando sempre é realizado pelo menos uma vez na rotina *do*...*while*.

Atenção: O *while* (...); a linha sempre termina com um ponto e vírgula.

Exemplo

```
H2 = 0;
H1 = 10;
do
{
  H2 = H2 +1;
  H1 = H1 -1;
} while ( H1 > 5 );
```

Os comandos dentro do bloco são cumpridos contanto que a H1 seja maior do que 5. A H2 recebe valor 4 quando a rotina pára.

Se a expressão permanece sempre TRUE, o resultado é uma rotina contínua:

```
do
  H2 = H2 +3;
while ( 1 );
```

Neste caso, a expressão tem o valor 1, que quer dizer que a rotina nunca é quebrada.

Esta rotina pode ser interrompida utilizando o comando *break*.

```
H2 = 0;
do
{
  H2 = H2 +3;
  if ( H2 > 20 )
    break;
} while ( 1 );
```

Neste exemplo, a rotina *do* é interrompida utilizando o comando *break* se o valor da variável IPOS$^{plus®}$ H2 for maior do que 20.

Também é possível um comando *continue*. Isto faz com que o programa pule para o final do bloco de comando e depois verifique a expressão.

```
H2 = 0;
do
{
  H2 = H2 +3;
  if ( H2 > 20 )
    break;
  if ( H2 > 10 )
    continue;
  ++H0;
} while ( 1 );
```

Neste exemplo, a incrementação da variável IPOS$^{plus®}$ H0 pára assim que o valor da variável IPOS$^{plus®}$ H2 for maior do que 10.

13.5 switch...case...default

Sintaxe

```
switch ( Expression )
{
  case value 1:  // Statement 1
                 break;
  case value 2:  // Statement 2
                 break;
     .
     .
  default:       // Statement n
}
```

O comando *switch* possibilita criar múltiplas derivações do programa dependendo do valor de uma expressão.

Se a Expression tem valor 1, Statement 1 é realizado, se a Expression tem valor 2, Statement 2 é realizado, etc. Se nenhum dos valores atribuídos a 'case' correspondem a 'Expression,' é executado o programa padrão *if* programado (Statement n).

Statements 1, 2,..., n são normalmente seqüências de comandos que terminam com um comando *break*. Se a seqüência dos comandos não termina com um comando *break*, todas as derivações *case* posteriores são realizadas até o comando *break* ser encontrado. O valor não é mais comparado com a expressão.

Statements 1, 2,..., n também podem ser funções de acesso. Por exemplo, um salto de programa pode ser configurado.

Value 1, Value 2, Value n devem ser constantes ou expressões constantes. Aqui não são permitidas variáveis.

A derivação padrão deve, se disponível, ser a última linha no comando *switch*.

```
switch ( H1 )
{
  case 1: ++H2;
          break;
  case 2: ++H3;
          break;
  default: ++H4;
           break;
}
```

Este programa extrai incrementos variável IPOS$^{plus®}$ H2 se o valor da variável IPOS$^{plus®}$ H1 for 1. Se seu valor for 2, então a variável IPOS$^{plus®}$ H3 é incrementada. A variável IPOS$^{plus®}$ H1 é incrementada dado algum outro valor da variável IPOS$^{plus®}$ H4.

Também é possível a variante a seguir:

```
switch ( H1 )
{
  case 1:
  case 2: ++H3;
          break;
  default: ++H4;
           break;
}
```

Este programa extrai incrementos variável IPOS$^{plus®}$ H3 se o valor da variável IPOS$^{plus®}$ H1 é 1 ou 2. A variável IPOS$^{plus®}$ H1 é incrementada dado algum outro valor da variável IPOS$^{plus®}$ H4.

13.6 return

A palavra-chave `return` finaliza o processamento de uma função e volta para o comando seguinte da função de acesso. O comando `return` possibilita finalizar funções prematuramente, por exemplo, aumentar a clareza de um programa C. No entanto, utilizar este comando com muita freqüência pode ter efeito contrário. Uma função deve conter o menor número de pontos de saída possível.

O exemplo a seguir mostra duas possibilidades de codificação para obter o mesmo resultado. O exemplo à esquerda utiliza o comando `return` para sair da função prematu-

ramente, ao passo que o exemplo à direita não utiliza `return`.

```
Function ()
{
  // Leave function when H1 is 5
  if ( H1 == 5 )
    return;
  H2=3;
}
```

```
Function ()
{
  // Skip statement when H1 = 5
  if (H1 != 5)
  {
    H2=3;
  }
}
```

14 Compiler – Funções

14.1 Funções definidas pelo usuário

O usuário pode programar funções (sub-programas). As funções definidas pelo usuário não podem ser acessadas quando os argumentos da função são transferidos. No entanto, isto também é desnecessário porque todas as variáveis são globais e não podem ser encapsuladas com variáveis locais. A estrutura da função é a seguinte:

```
FunctionName()
{
  // Statements
}
```

Os protótipos da função no cabeçalho do programa não têm que ser definidos, ao contrário do ANSI-C.

Uma vez a função ter sido realizada, é processada a linha seguinte à função de acesso.

A função pode ser cancelada prematuramente utilizando o comando *return*. Neste caso, o processamento do programa é continuado com a linha seguinte à função de acesso.

Utilizando o comando *return* pode aumentar a clareza da estrutura do programa. No entanto, utilizar este comando com muita freqüência pode ter efeito contrário. Uma função deve conter o menor número de pontos de saída possível.

```
SampleFunction()
{
  // Leave function when H1 is 5
  if ( H1 == 5 )
    return;
  H2=3;
}
```

A função definida pelo usuário não pode ser acessada de vários tasks.

14.2 Resumo dos comandos para funções padrão

Uma grande parte dos comandos IPOS$^{plus®}$ da máquina, que são familiares a linguagem assembler IPOS$^{plus®}$, são reproduzidos na linguagem de alto nível do compilador IPOS$^{plus®}$ na forma de certas construções sintáticas. Por exemplo, os comandos aritméticos (ADD, SUB, etc.) são criados por operadores adequados (+, -, etc.) ou os comandos ajustados (SET...) são substituídos pelo operador de programação (=). No entanto, há também comandos com nenhum equivalente na linguagem de programação. Estes comandos (GOA, BSET, etc.) são reproduzidos utilizando funções que fazem parte do compilador e são portanto descritos como funções padrão, ao contrário das funções definidas pelo usuário.

Os parâmetros dos comandos IPOS$^{plus®}$ da máquina tornam-se argumentos das funções padrão. Os nomes de todas as funções padrão iniciam com (_), deste modo é mais fácil distingüí-las das funções do usuário no texto fonte.

As constantes especificadas como argumento em muitas funções são definidas no arquivo de cabeçalho CONST.H (MOVIDRIVE® A) / CONSTB.H (MOVIDRIVE® B). Se você preferir utilizar seus próprios nomes, você pode definí-los utilizando a diretiva #define.

Funções padrão do bit

Comando	Função	Referência
_BitClear	Muda para nível 0 um bit dentro de uma variável	página 189
_BitMove	Copia um bit em uma variável para um bit em outra variável.	página 190
_BitMoveNeg	Copia um bit em uma variável para um bit em outra variável e o nega.	página 190
_BitSet	Muda para nível 1 um bit dentro de uma variável	página 190

Funções padrão de comunicação

Comando	Função	Referência
_MoviLink	Processa e/ou amplia os dados de parâmetro via RS-485 ou system bus.	página 201
_MovCommDef	Transfere os dados de processo via RS-485 (Especialmente com MQx para MOVIMOT®).	página 206
_MovCommOn	Inicia a transferência dos dados de processo via RS-485.	página 208
_SBusCommDef	Define a ampliação dos dados de processo via system bus.	página 208
_SBusCommOn	Inicia a transferência dos dados de processo via system bus.	página 213
_SBusCommState	Inicia a transferência dos dados de processo via system bus (somente para MOVIDRIVE® B).	página 214

Funções padrão de posicionamento

Comando	Função	Referência
_Go0	Realiza o referenciamento	página 197
_GoAbs	Posicionamento absoluto	página 198
_GoRel	Posicionamento relativo	página 199

Funções padrão do programa

Comando	Função	Referência
_InputCall	Chama uma função definida quando bits específicos selecionados estão em nível 1 ou nível 0 nos bornes de entrada.	página 200
_Nop	Sem operação	página 208
_SystemCall	Chama uma função definida quando o sistema ocorre mesmo.	página 220
_SetTask	Define uma função como task 2 ou task 3 e a inicia ou pára (somente MOVIDRIVE® B).	página 217
_SetTask2	Define uma função como task 2 e a inicia ou pára (administrada por MOVIDRIVE® B).	página 217
_Wait	Espera por um período específico	página 221
_WaitInput	Espera até um certo nível estar presente em certos bornes de entrada.	página 222
_WaitSystem	Espera até ocorrer um evento do sistema.	página 222

Funções padrão de ajuste

Comando	Função	Referência
_Copy	Bloco-a-bloco, cópia consistente das variáveis.	página 191
_GetSys	Lê um valor interno do sistema.	página 192
_SetInterrupt	Define uma função como rotina de interrupção e a ativa ou desativa.	página 214
_SetVarInterrupt	Define uma função como rotina variável e a ativa ou desativa (somente MOVIDRIVE® B).	página 218
_SetSys	Ajusta um valor interno do sistema.	página 215

Funções padrão especiais do conversor

Comando	Função	Referência
_AxisStop	O conversor é parado.	página 189
_FaultReaction	Ajusta a reação de irregularidade para uma irregularidade selecionada.	página 191
_Memorize	Salva ou carrega variáveis ou programa IPOS$^{plus®}$.	página 201
_TouchProbe	Libera ou bloqueia uma entrada touch probe.	página 221
_WdOn	Ajusta o timer Watchdog para um valor específico.	página 223
_WdOff	Desliga o Watchdog.	página 223

14.3 Funções padrão

Este capítulo indica as funções padrão em ordem alfabética. Isto facilita encontrar a função padrão que você está procurando.

_AxisStop

Sintaxe

```
_AxisStop( type )
```

Descrição

O eixo do conversor é parado quando a palavra de controle IPOS$^{plus®}$ é escrita. Uma repartida deve ser cumprida pela função liberação através da palavra de controle IPOS$^{plus®}$. O argumento pode ser utilizado para especificar o tipo de parada do eixo, ou a desativação do bloqueio pode ser especificada através da palavra de controle IPOS$^{plus®}$.

Argumento

tipo

AS_RSTOP	Frenagem com a rampa de parada rápida, seguida pelo estado "No enable." A última posição final (H492) transmitida é retida. O bloqueio através da palavra de controle (comando ASTOP (IPOS ENABLE) é necessário antes do comando de deslocamento posterior). O freio é atuado se a função freio estiver ativada. A mensagem "In position" é ajustada.
AS_HCTRL	Frenagem com a rampa da unidade básica P131/P133 seguida pelo controle de posição; a última posição final transmitida é retida. O bloqueio através da palavra de controle (o comando ASTOP (IPOS ENABLE) é necessário antes do comando de deslocamento). O freio não é atuado se a função freio estiver ativada.
AS_PSTOP	Posicionamento de parada com a rampa de posicionamento P911 / P912 e calculada a posição final "STOP" (possível somente no modo de posicionamento), seguida pelo controle de posição. A última posição final (H492) transmitida é regravada pela posição de parada. Sem bloqueio através da palavra de controle (nenhum comando ASTOP (IPOS ENABLE) necessário antes do comando de deslocamento posterior). O freio não é atuado se a função freio estiver ativada. Obs: Desde que a posição atual seja utilizada como posição de ajuste em parada, o comando não pode ser processado ciclicamente. Este é o caso em eixos com forças de processo ou elevações porque caso contrário o eixo desvia lentamente de posição.
AS_ENABLE	O bloqueio é cancelado utilizando a palavra de controle IPOS$^{plus®}$.

Desde que a posição atual seja utilizada como posição de ajuste em parada, o argumento AS_PSTOP não pode ser processado ciclicamente. Este é o caso em eixos com forças de processo ou elevações, porque caso contrário o eixo desvia lentamente de posição.

Exemplo

```
main()
{
  _GoAbs( GO_NOWAIT,3000 ); // Start travel request
  _AxisStop( AS_PSTOP );    // Cancel travel request
                            // Statements during standstill
  _AxisStop( AS_ENABLE );   // Revoke inhibit
  _GoAbs( GO_NOWAIT,3000 ); // Send new travel request
}
```

_BitClear

Sintaxe

```
_BitClear( h , bit )
```

Descrição

_BitClear ajusta o bit dentro da variável h para nível lógico zero.

Argumentos

h Nome da variável

bit Expressão constante para a posição do bit

Exemplo

```
main()
{
  _BitClear( H100, 3 ); // Deletes bit 3 in H100
}
```

_BitMove

Sintaxe

```
_BitMove( h2 , bit2, h1, bit1 )
```

Descrição

Copia o bit com o número bit1 na variável IPOS$^{plus®}$ h1 para o bit com o número bit2 na variável IPOS$^{plus®}$ h2. Todos os bits da h1 e todos os outros bits da h2 permanecem inalterados. As posições do bit de uma variável são numeradas de 0 até 31. O bit menos significativo tem o número 0.

Argumentos

h2 Nome da variável final

bit2 Número do bit final

h1 Nome da variável fonte

bit1 Número do bit fonte

Exemplo

```
main()
{
  _BitMove( H1, 3,H2, 4 ); // Copies H1.3 = H2.4
  _BitMove( H1, 1,H1, 0 ); // Copies H1.1 = H1.0
}
```

_BitMoveNeg

Sintaxe

```
_BitMoveNeg( h2 , bit2, h1, bit1 )
```

Descrição

Copia o bit com o número bit1 na variável IPOS$^{plus®}$ h1 para o bit com o número bit2 na variável IPOS$^{plus®}$ h2. O bit é negado durante este processo. Todos os bits da h1 e todos os outros bits da h2 permanecem inalterados. As posições do bit de uma variável são numeradas de 0 até 31. O bit menos significativo tem o número 0.

Argumentos

h2 Nome da variável final

bit2 Número do bit final

h1 Nome da variável fonte

bit1 Número do bit fonte

Exemplo

```
main()
{
  _BitMoveNeg( H1, 3,H2, 4 ); // Copies H1.3 = NOT (H2.4)
  _BitMoveNeg( H1, 1,H1, 0 ); // Copies H1.1 = NOT (H1.0)
}
```

_BitSet

Sintaxe

```
_BitSet( h , bit )
```

Descrição

Dentro da variável IPOS$^{plus®}$ h, _BitSet ajusta o bit com o number bit para um.

Argumentos

h Nome da variável

bit Expressão constante com o número do bit a ser ajustado

Exemplo

```
main()
{
  _BitSet( H100, 3 ); // Sets Bit 3 in H100
}
```

_Copy

Sintaxe

```
_Copy ( h2 , h1, no. )
```

Descrição

Copia o número no. para variáveis consecutivas como um bloco variável. h1 especifica o nome da primeira variável fonte, h2 o nome da primeira variável final. Podem ser copiadas no máximo 10 variáveis.

Argumentos

h2 Nome da primeira variável final

h1 Nome da primeira variável fonte

no Expressão constante para o número de variáveis IPOS$^{plus®}$ para cópia

Exemplo

```
main()
{
  _Copy( H1,H5, 3 ); // Copy H1 = H5, H2 = H6, H3 = H7
}
```

_FaultReaction

Sintaxe

```
_FaultReaction( fnr, r )
```

Descrição

Este comando pode ser utilizado para programar a reação do sistema para uma irregularidade do conversor. Como tal, o comando deve ser cumprido antes de ocorrer a irregularidade. O argumento especifica a irregularidade e a reação correspondente se esta irregularidade ocorrer.

Todas as reações de irregularidade na lista de irregularidade nas instruções de operação ou no manual de sistema que têm um ponto na coluna P, podem ser programadas como reações de irregularidade.

Argumentos

fno Expressão constante para o número da irregularidade (ver lista de irregularidades nas instruções de operação)

r Expressão constante para uma reação de irregularidade que pode adotar os seguintes valores:

FR_NORESP: Sem reação; o erro não é indicado.
FR_DISPLAY: O erro somente é indicado; o conversor continua a funcionar.
FR_SWOFF_F: Bloqueio do estágio de saída e bloqueio do conversor. Reativação desejada.
FR_ESTOP_F: Pára na rampa de parada de emergência com bloqueio do conversor. Reativação desejada.
FR_RSTOP_F: Pára na rampa de parada rápida com bloqueio do conversor. Reativação desejada.
FR_SWOFF_W: Bloqueio do estágio de saída sem bloqueio do conversor. Reativação re-libera o conversor.
FR_ESTOP_W: Pára na rampa de parada de emergência sem bloqueio do conversor. Reativação re-libera o conversor.
FR_RSTOP_W: Pára na rampa de parada rápida sem bloqueio do conversor. Reativação re-libera o conversor.

FR_SWOFF_F / FR_ESTOP_F / FR_RSTOP_F: O conversor é reinicializado; isto é, o IPOS$^{plus®}$ é reiniciado.

FR_SWOFF_W / FR_ESTOP_W / FR_STOP_W: O conversor não é reinicializado; isto é, o IPOS$^{plus®}$ continua a funcionar.

Exemplo

```
main()
{
    _FaultReaction( 26,FR_SWOFF_F ); // Emergency stop / Malfunction
with ext. fault
}
```

_GetSys

Sintaxe `_GetSys( h, sys )`

Descrição Carrega o valor interno do sistema em uma ou mais variáveis IPOS$^{plus®}$.

Argumentos

h Nome da variável final ou estrutura final

sys Expressão que designa o valor do sistema. sys pode adotar um dos seguintes valores:

GS_ACTCUR: Corrente ativa em 0.1 % da corrente nominal do conversor
GS_ACTSPEED: Rotação atual em 01 rpm
GS_SPSPEED: Rotação de ajuste em 0.1 rpm
GS_ERROR: Código da irregularidade de acordo com a tabela "Mensagens de irregularidade e lista de irregularidades" no manual de sistema
GS_SYSSTATE: Valor do display de 7 segmentos de acordo com a tabela "Indicação do modo de operação" no manual de sistema
GS_ACTPOS: Posição atual dependendo do encoder selecionado em P941 (H509, H510 ou H511)
GS_SPPOS: Posição de ajuste H491
GS_TPOS: Posição final do gerador de perfil
GS_INPUTS: Entradas digitais H483 (MOVIDRIVE® A) / H520 (MOVIDRIVE® B) da unidade básica e opcionais
GS_DEVSTATE: Idêntico a palavra de estado 1 do protocolo fieldbus (código da irregularidade e estado de operação)
GS_OUTPUTS: Saídas digitais H482 (MOVIDRIVE® A) / H522 (MOVIDRIVE® B) da unidade básica e opcionais
GS_IxT: Utilização em 0.1 % da corrente nominal do conversor

GS_ACTPOS / GS_SPPOS / GS_TPOS: A resolução depende do encoder selecionado em P941:
– Encoder do motor: 4096 Inc./volta
– Encoder externo X14: Resolução do encoder P944
– DIP (Encoder SSI): Resolução do encoder P955

GS_ANINPUTS: Valor de tensão / valor de corrente das entradas analógicas 1 e 2
Entrada de tensão: –10 V ... 0 ... +10 V = –10000 ... 0 ... 10000
Entrada de corrente: 0 ... 20 mA = 0 ... 5000 / 4 ... 20 mA = 1000 ... 5000
– h +0 = Entrada analógica 1
– h + 1 = Entrada analógica 2

GS_CAM: Utilizado para execução de um controlador came
– Com o comando GETSYS, pode ser utilizado por conversor, um controlador came padrão com 4 saídas. Para o novo MOVIDRIVE®, é disponível um controlador came expandido com 8 saídas (MDx_A como da versão .14 / MCH como da versão .13 / MDx_B).
– Hxx é a primeira variável de uma estrutura de dados (CamControl ou GS_CAM). O bit com maior importância (bit 31) é utilizado na Hxx para decidir qual controlador came o comando GETSYS refere-se.
– Bit 31 = 0: Controlador came padrão (todos MOVIDRIVE®). O comando GETSYS ativa o controlador came. Os cames são formados depois do comando GETSYS ser processado. Se o controlador came é para trabalhar ciclicamente, o comando deve ser acessado ciclicamente.
– Bit 31 = 1: Controlador came expandido (somente MDx_A como da versão .14 / MCH como da versão .13 / MDx_B com versão aplicação e modo de operação CFC ou SERVO). O comando GETSYS ativa o controlador came, os cames são formados ciclicamente no fundo.
– Para mais informação sobre os controladores came e a estrutura de dados, consultar a seção "Controladores came" no capítulo "Detecção da Posição e Posicionamento".

GS_ANOUTPUTS: Saídas analógicas opcionais, onde –10 V ... 0 ... +10 V = –10000 ... 0 ... 10000.
– h = Saída analógica 1
– h + 1 = Saída analógica 2

GS_TIMER0: Valor do contador do TIMER 0 H489 em ms
GS_TIMER1: Valor do contador do TIMER 1 H488 em ms

GS_PODATA: Lê o buffer de dados PO. Independente do número de itens de dados PO, 2 itens de dados PO ou 10 itens de dados PO são lidos (dados enviados do mestre para o conversor).
– h + 0: Bus type
  0 = Reserved
  1 = S0 (RS485 #1)
  2 = S1 (RS485 #2)
  3 = Fieldbus
  4 = Reserved
  5 = SBus
  8 = SBus 2 (somente MOVIDRIVE® B)
– h + 1 = Number of PO data items
– h + 2 = PO1
– h + 3 = PO2
– h + 4 = PO3
– h + 5 = PO4
– h + 6 = PO5
– h + 7 = PO6
– h + 8 = PO7
– h + 9 = PO8
– h + 10 = PO9
– h + 11 = PO10

GS_DCVOLT: Tensão do circuito intermediário CC [V]
GS_RELTORQUE: Torque relativo. O valor é disponível nos modos de operação CFC... e SERVO... .
GS_RELTORQUEVFC: O torque relativo é o valor indicado, baseado na corrente nominal do conversor para o torque no eixo de saída do motor, em 0.1 % da corrente nominal do conversor. O torque absoluto pode ser calculado a partir deste valor utilizando a seguinte fórmula:
$M_{abs} = M_{rel} \times I_N \times M_N / 1000 / I_{QN}$
M_{abs} = Torque absoluto
I_N = Corrente nominal do conversor
M_{rel} = Torque relativo baseado em 0.1 % I_N
M_N = Torque nominal do motor [Nm]
I_{QN} = Corrente nominal Q [A] para o tipo de conexão selecionado. O valor é disponível nos modos de operação CFC e SERVO / VFC1, VFC1 & hoist, VFC1 & DC braking e VFC1 & flying start.

GS_ACTSPEEDEXT: Rotação atual do encoder externo (X14)
– h = Time base, filtro de valor médio para detecção da rotação do encoder externo.
  Faixa de ajuste: 5 ms ... 31 ms
– h + 1 = Tipo do encoder
  – 0 = Encoder X14,
  – 1 = Encoder DIP
– h + 2 = Numerador para a escala da faixa de valor do usuário: -2^{15} ... 0 ... $+(2^{15} - 1)$
– h +3 = Denominador para a escala da faixa de valor do usuário: 1 ... $(2^{15} - 1)$
– h + 4 = DPointer, indicador do resultado da variável H', onde H' = resultado; conversor: [nX14] = (Inc/time base)

Exemplo: Inserir a rotação em voltas por hora. A estrutura `GS_ACTSPEEDEXT gLAActSpeed;` foi definida para este processo.

```
gLAActSpeed.TimeBase = 30;      // Average value filter 30 ms
gLAActSpeed.EncType = 0;        // Encoder is connected to X14
gLAActSpeed.Numerator = -11250; // Conversion into arcs per hour 11250 / 384
gLAActSpeed.Denominator = 384;  // = (1000 ms x 60 s x 60 min) / (Inc. x Time base)
gLAActSpeed.DPointer = numof(hArcsPerHour); // -11250 negated representation

_GetSys(gLAActSpeed,GS_ACTSPEEDEXT);
```

SPEEDMONITOR
Valor do contador da função de controle da rotação
O comando GETSYS pode ser utilizado como um pré-aviso para o controle da rotação. O controle da rotação é ativado quando a corrente está no limite atual para o número de segundos especificado em P501. Por exemplo, se P501 = 200 ms, o comando GETSYS pode ser utilizado para verificar o valor do contador. A rotação de deslocamento é reduzida após 50 ms. Deste modo, o retorno do deslocamento pode ser feito em rotação rápida e, quando sob carga, a rotação pode ser reduzida automaticamente pelo conversor.

As estruturas padrão SEW a seguir são disponíveis para o comando _GetSys:

Comando	Estrutura padrão	Elementos	Descrição breve
_GetSys	GSAINPUT	Input1	Valor de tensão da entrada analógica 1
		Input2	Valor de tensão da entrada analógica 2
	GSAOUTPUT	Output1	Valor de tensão para saída analógica opcional 1
		Output2	Valor de tensão para saída analógica opcional 2
	GSCAM	SourceVar	Número da variável a qual o comando é realizado
		DbPreCtrl	Pré-controle do tempo de atraso em 0.1 ms
		DestVar	Número da variável que deve receber o resultado
		BitPosition	Posição do bit na variável de resultado
		BitValue	Polaridade na variável de resultado
		NumOfCam	Número de blocos came (máx. 4)
		PosL1	Valor limite antihorário do bloco came 1
		PosR1	Valor limite horário do bloco came 1
		PosL2	Valor limite antihorário do bloco came 2
		PosR2	Valor limite horário do bloco came 2
		PosL3	Valor limite antihorário do bloco came 3
		PosR3	Valor limite horário do bloco came 3
		PosL4	Valor limite antihorário do bloco came 4
		PosR4	Valor limite horário do bloco came 4
	GSCAM_EXT	CamControl	O bit 2^{31} sempre deve ser ajustado. 0x8000 0000 = função inativa, nenhuma saída de came nova será gerada, o ajuste das saídas será retido e apagado somente após um reset ou quando a tensão é desligada/ligada. 0x8000 0001 = função ativa internamente, mas todas as saídas de came são desligadas 0x8000 0002 = função ativa se o conversor é referenciado (H473, Bit20 =1) 0x8000 0003 = função ativa mesmo sem o conversor referenciado
		CamReserved1	Reservado
		CamOutShiftLeft	Desloca o buffer de dados interno das saídas por n dígitos à esquerda antes de escrever para a variável final CamDestination. Importante: O processo de deslocamento apaga a informação das saídas superiores. Isto significa que se o fator de deslocamento for 3, as 3 saídas superiores com duração do ciclo de 4 ms não são mais disponíveis, e as 4 saídas com duração do ciclo de 1 ms são atribuídas aos bits 3-6 e a saída com duração do ciclo de 4 ms é atribuída ao bit 7.
		CamForceOn	Máscara para ajustar as saídas obrigatórias; a máscara atinge o buffer de dados interno antes de deslocar com CamOutShiftLeft (NÃO a variável final definida com CamDestination)
		CamForceOff	Máscara para apagar as saídas obrigatórias; a máscara atinge o buffer de dados interno antes de deslocar com CamOutShiftLeft (NÃO a variável final definida com CamDestination) CamForceOff domina CamForceOn

Comando	Estrutura padrão	Elementos	Descrição breve
		CamSource	Bit 2^{31} comuta entre as variáveis de referência pré-definidas e um indicador para a variável de referência aleatória. Bit 2^{31} = 0: • 0 = encoder X15 (encoder do motor, H511) • 1 = encoder X14 (encoder externo, H510) • 2 = encoder H509 (encoder absoluto DIP11A) • 3 = encoder virtual • todos os valores seguintes são reservados! Bit 2^{31} = 1: CamSource inclui um indicador para uma variável IPOS$^{plus®}$ $+2^{31}$
		CamDestination	Indicador para a variável final. Os bits não utilizados nas variáveis finais são disponíveis para outras funções (se deslocar as saídas por quatro à esquerda com Shift Left, isto libera até os bits 0-3, os bits 4-7 são disponíveis para as funções came e os bits 8-31 são disponíveis para qualquer programação. Se as saídas came são atribuídas às saídas do conversor (por ex. H481), você tem que reservar estas saídas digitais com P620 – P639 como saídas IPOS$^{plus®}$. Os bits não utilizados nesta palavra são disponíveis para outras saídas.
		CamOutputs	Número de saídas (máx. 8)
		CamData 1	Indicador para primeira estrutura CamOutput (1ª saída)
		...	
		CamData 8	Indicador para última estrutura CamOutput (8ª saída)
	CAM_EXT_OUT	DeadTime	Compensação do tempo de atraso para este canal (–500 ms..0..+500 ms) para compensar o tempo de atraso de um atuador conectado ao conversor. A saída é pré-definida, dependendo da taxa de mudança do valor da variável de referência, de tal modo que a saída é comutada com antecedência por este intervalo de tempo.
		CamAreas	Número das janelas de posição para este canal (1 ... 4); o valor limite antihorário sempre deve ser menor do que o horário. Se um módulo de eixo necessita uma janela de posição que exceda o limite 360° - 0°, então esta faixa terá que ser dividida em duas janelas de posição. Este processo deixa o operador ajustar três faixas relacionadas a esta saída.
		LeftLimit1	Limite antihorário, janela 1
		RightLimit1	Limite horário, janela 1
		...	...
		LeftLimit4	Limite antihorário, janela 4
		RightLimit4	Limite horário, janela 4
	GSPODATA3	Bus types	0 = Reservado 1 = S0 (RS485 #1) 2 = S1 (RS485 #2) 3 = Fieldbus 4 = Reservado 5 = SBus
		Len	Número de itens dos dados de saída de processo
		PO1	Dados de saída de processo 1
		PO2	Dados de saída de processo 2
		PO3	Dados de saída de processo 3

Comando	Estrutura padrão	Elementos	Descrição breve
	GSACTSPEEDEXT	TimeBase	Duração do ciclo para detecção da rotação do encoder externo, faixa de ajuste: 5 ms ... 31 ms
		EncType	0 = Encoder X14,1 = Encoder DIP
		Numerator	Numerador para escala do usuário Faixa de valor: -2^{15} ... 0 ... $+(2^{15}-1)$
		Denominator	Denominador para escala do usuário Faixa de valor: 1 ... $(2^{15}-1)$
		DPointer	Indicador para a variável de resultado H'
	GSPODATA10	Bus types	0 = Reservado 1 = S0 (RS485 #1) 2 = S1 (RS485 #2) 3 = Fieldbus 4 = Reservado 5 = SBus
		Len	Número de itens dos dados de saída de processo
		PO1	Dados de saída de processo 1
		PO2	Dados de saída de processo 2
		PO3	Dados de saída de processo 3
		PO4	Dados de saída de processo 4
		PO5	Dados de saída de processo 5
		PO6	Dados de saída de processo 6
		PO7	Dados de saída de processo 7
		PO8	Dados de saída de processo 8
		PO9	Dados de saída de processo 9
		PO10	Dados de saída de processo 10

Exemplo

```
#include <const.h>

GSAINPUT Ain;

main()
{
_GetSys( Ain,GS_ANINPUTS ); // Read in analog inputs in structure Ain
}
```

_Go0

Sintaxe

```
_Go0( type )
```

Descrição

Este comando ativa o referenciamento do eixo. O argumento define o tipo de referenciamento. O referenciamento é ajustado com P903 e somente pode ser mudado aqui.

Argumentos

type Expressão para ajuste do tipo de deslocamento durante o referenciamento. type pode adotar um dos seguintes valores:

GO0_C_W_ZP
GO0_U_W_ZP
GO0_C_NW_ZP
GO0_U_NW_ZP
GO0_C_W_CAM
GO0_U_W_CAM
GO0_C_NW_CAM
GO0_U_NW_CAM
GO0_RESET

O significado das letras individuais é conforme a seguir:

C (Conditional) =	Referenciamento somente se ainda não tiver sido realizado
U (Unconditional) =	Referenciado sempre, independente se o eixo já foi ou não referenciado
W (Wait) =	Espera nesta linha de comando até o referenciamento ser realizado
NW (NoWait) =	Processa a próxima linha de comando durante o referenciamento (recomendação)
ZP (Zero Pulse) =	Referenciamento para pulso zero
CAM =	Referenciamento para came de referência
RESET =	Referenciamento que iniciou é interrompido e o acesso é reativado. Um eixo que foi referenciado é agora não referenciado.

ZP e CAM não tem efeito se o tipo de referenciamento P903 for ajustado para 0 ou 5. Se o tipo de referenciamento P903 for ajustado para tipo 3 ou 4, CAM não pode ser ajustado.

Exemplo

```
main()
{
  _Go0( GO0_C_W_ZP ); // Reference waiting for a zero pulse
}
```

_GoAbs

Sintaxe `_GoAbs( type, pos )`

Descrição Posicionamento absoluto para a posição especificada.

A mensagem "IPOS in position" é atualizada dentro do comando GOA ou GOR; isto é, a mensagem pode ser verificada diretamente na próxima linha do programa.

Argumentos **type** Expressão para o tipo do comando de movimento. type pode adotar um dos seguintes valores:

GO_NOWAIT: Sem espera, recomeça o processamento do programa na próxima linha do comando, imediatamente após enviar o comando de movimento (recomendação)
GO_WAIT: Espera nesta linha de comando até o deslocamento ser completado

pos Contém a posição final absoluta; o can seguinte representa pos:

Expressão constante para posição final
Nome de uma variável contendo a posição final
Nome de uma variável indireta

Ajustes de parâmetro para todos os comandos de posicionamento

Parâmetros	Explicação
P913 / P914	Rotações de deslocamento (podem ser mudadas no programa utilizando SETSYS).
P911 / P912	Rampas de posicionamento (aceleração) (podem ser mudadas no programa utilizando SETSYS).
P915 / P203	Freqüências fixas que podem ser utilizadas para influenciar os trancos.
P933	Suavização de picos (somente com MOVIDRIVE® B).
P916	Tipo de rampa.
P917	Modo de rampa.

Exemplo

```
// Standard structures for speed and ramp
SSPOSSPEED tPosSpeed;
SSPOSRAMP tPosRamp;

main()
{
  // Set speed and ramp
  tPosSpeed.CW = tPosSpeed.CCW = 1000 * 10; // Speed 1000 1/min
  tPosRamp.Up = tPosRamp.Down = 1000;       // Ramp is based on 3000 1/min
  _SetSys(SS_POSRAMP, tPosRamp);

  _SetSys(SS_POSSPEED, tPosSpeed);
  // If the speed and ramp are not changed in the program,
  // the values in SHELL apply / see the table

  _GoAbs (GO_WAIT, 3000);                   // Moves to position 3000 In.
}
```

Se a função módulo é utilizada para posicionamento, os comandos GOA e GOR não podem ser utilizados. A posição final é escrita diretamente para H454.

_GoRel

Sintaxe

```
_GoRel( type, pos )
```

Descrição

Posicionamento relativo para a distância baseada na posição atual.

A mensagem "IPOS in position" é atualizada dentro do comando GOA ou GOR; isto é, a mensagem pode ser verificada diretamente na próxima linha do programa.

Argumentos

type Expressão para o tipo do comando de movimento. type pode adotar um dos seguintes valores:

GO_NOWAIT: Sem espera, recomeça o processamento do programa na próxima linha do comando, imediatamente após enviar o comando de movimento (recomendação)
GO_WAIT: Espera nesta linha de comando até o deslocamento ser completado

pos Contém a distância relativa; o can seguinte representa pos:

Expressão constante para distância
Nome de uma variável contendo a distância
Nome de uma variável indireta

Exemplo

```
// Standard structures for speed and ramp
SSPOSSPEED tPosSpeed;
SSPOSRAMP tPosRamp;

main()
{
  // Set speed and ramp
  tPosSpeed.CW = tPosSpeed.CCW = 1000 * 10; // Speed 1000 1/min
  tPosRamp.Up = tPosRamp.Down = 1000;       // Ramp is based on 3000 1/min
  _SetSys(SS_POSRAMP, tPosRamp);

  _SetSys(SS_POSSPEED, tPosSpeed);
  // If the speed and ramp are not changed in the program,
  // the values in SHELL apply / see the table

  _GoAbs (GO_WAIT, 3000);                   // Moves to position 3000 In.
}
```

Se a função módulo é utilizada para posicionamento, os comandos GOA e GOR não podem ser utilizados. A posição final é escrita diretamente para H454.

_InputCall

Sintaxe

```
_InputCall ( level, mask, function name )
```

Descrição

A função é utilizada para acessar uma função definida pelo usuário dependendo do nível presente nos bornes de entrada. O nome da função, a polaridade desejada do nível de entrada e os bornes relevantes são especificados como argumentos. A função do evento é acessada quando todos os bornes de entrada marcados com um na máscara tenham nível "1" (level = IC_HIGH) ou nível "0" (level = IC_LOW).

Argumentos

level Expressão constante que especifica o nível de sinal que os bornes são testados. Esta expressão pode adotar um dos seguintes valores:

IC_HIGH: HIGH level ("1" level)
IC_LOW: LOW level ("0" level)

mask Expressão binária constante que especifica os bornes a ser testados.

Os bits na expressão têm o seguinte significado

Bit 0: DI00, mask = 0b1
Bit 1: DI01, mask = 0b10
Bit 2: DI02, mask = 0b100
Bit 3: DI03, mask = 0b1000
Bit 4: DI04, mask = 0b10000
Bit 5: DI05, mask = 0b100000
Bit 6: DI10, mask = 0b1000000
Bit 7: DI11, mask = 0b10000000
Bit 8: DI12, mask = 0b100000000
Bit 9: DI13, mask = 0b1000000000
Bit 10: DI14, mask = 0b10000000000
Bit 11: DI15, mask = 0b100000000000
Bit 12: DI16, mask = 0b1000000000000
Bit 13: DI17, mask = 0b10000000000000
Bit 14-31: reserved

Pode ser selecionada uma combinação de entrada ajustando os bits apropriados na máscara para 1. Por exemplo, para verificar DI00 e DI03, a máscara deve ser: 0b1001

function name Nome da função do evento. (Importante: Ao contrário da função call, somente o nome da função sem () é especificado aqui)

Exemplo

```
#include <constb.h>

#define DI02 0b100 // DI02 = 0b100

TerminalIsOne ()
{
  // Statements of the event function
}

main()
{
  while(1)
  {
    // Main program loop task 1
    _InputCall( IC_HIGH,DI02,TerminalIsOne );
    // if terminal DI02 == HIGH ("1"), call the function
  }
}
```

_Memorize

Sintaxe

```
_Memorize( action )
```

Descrição

Permite que os programas IPOS$^{plus®}$ e/ou as variáveis sejam salvos ou carregados, na ou a partir da, memória não-volátil (EEPROM) no conversor. A ação é especificada através do argumento.

Argumentos

action Expressão constante para ação. action pode adotar um dos seguintes valores:

MEM_NOP: Nenhum dado é salvo
MEM_STALL: Salva o programa e as variáveis
MEM_LDALL: Carrega o programa e as variáveis
MEM_STPRG: Somente salva o programa
MEM_LDPRG: Somente carrega o programa
MEM_STDATA: Somente salva as variáveis
MEM_LDDATA: Somente carrega as variáveis

Quando utilizar o comando _Memorize(), observe que as variáveis armazenadas nas variáveis não-voláteis H0 127 e todos os parâmetros não são escritos ciclicamente. Isto é porque o número de operações com o meio de armazenamento EEPROM é restrito a 10^5 operações de armazenamento.

Exemplo

```
main()
{
  _Memorize( MEM_STDATA ); // Save variables H0 ... H127 to EEPROM
}
```

_MoviLink

Sintaxe

```
_MoviLink( h )
```

Descrição

O comando MOVLNK permite que sejam feitas mudanças consideráveis nos parâmetros do conversor e em qualquer outra unidade que possa ser conectada através do system bus ou da RS-485. Para garantir a segurança das pessoas e dos sistemas, tomar especial cuidado quando mudar os parâmetros do conversor. Em todos os casos, precauções de segurança de nível mais alto devem ocorrer para neutralizar todos os erros possíveis de programação.

Quando o comando é acessado, o MOVLNK lê e escreve uma vez os dados de processo, as variáveis ou os parâmetros de um conversor a outro, ou lê ou escreve uma vez as variáveis ou os parâmetros dentro de um conversor.

Os parâmetros são lidos / escritos utilizando o endereçamento index. Os respectivos números index são listados no manual "Fieldbus Unit Profile with List of Parameters". Os números index também podem ser indicados no SHELL selecionando o parâmetro e pressionando a combinação das teclas <Ctrl>+<F1>.

Pode ser utilizada uma interface SBus ou RS-485 para comunicação entre 2 conversores.

O MOVILINK pode ser utilizado, por exemplo, para salvar as variáveis de um contador de quantidade dentro de um conversor protegido contra falha de potência, sem salvar a faixa inteira de falha de potência utilizando o comando MEM. Os dados de processo não podem ser ampliados dentro de um conversor utilizando o comando MOVLINK.

Utilizar o acesso index através do MOVILINK para escrever / ler valores em IPOS$^{plus®}$ que não podem ser obtidos com GETSYS / SETSYS. Este acesso possibilita o conversor configurar-se no segmento de inicialização.

Antes do comando ser acessado, inicializar as variáveis que o comando utiliza (estrutura do comando). O começo desta estrutura de comando é transferido para o comando como um argumento. A estrutura de dados contém dados a ser escritos ou lidos.

Ajustar os parâmetros para comunicação no emissor (mestre) e no receptor (escravo). O comando MOVILINK somente é acessado no emissor (mestre).

Os dados de processo somente podem ser ampliados entre o MOVIDRIVE® e o MOVIMOT® através da RS-485. Neste processo o MOVIDRIVE® sempre é o emissor e o MOVIMOT® sempre é o receptor.

Características	RS-485	SBus
Tempo de ciclo	30 ms	10 ms (5 ms, somente PD)
Emissor – receptor	Sim	Sim
Emissores múltiplos[1)]	Não	Sim
Comunicação com MOVIMOT®	Sim (somente PD, MOVIMOT® é o receptor)	Não
Observar	Não utilizar a interface Xterminal	Resistores de terminação da rede em ambos os lados do SBus

1) Vários dos conversores conectados podem iniciar a comunicação

Argumentos

h Iniciando a variável de estrutura do comando

A estrutura do comando é configurada conforme a seguir:

h + 0: Tipo de comunicação
- ML_BT_S0: S0 (RS485 #1 = Terminal)
- ML_BT_S1: S1 (RS485 #2 = X13 na placa de controle)
- ML_BT_SBUS: SBus

h +1: Endereço simples ou de grupo do conversor final a ser endereçado

h +2: Especificação dos canais de processo (PD) e de parâmetro (PARAM) para transferência de dados:
- ML_FT_PAR1: PARAM+1PD
- ML_FT_1: 1PD
- ML_FT_PAR2: PARAM+2PD
- ML_FT_2: 2PD
- ML_FT_PAR3: PARAM+3PD
- ML_FT_3: 3PD
- ML_FT_PAR: PARAM (sem PD)

h +3: Serviço de comunicação
- ML_S_RD: Serviço de leitura
- ML_S_WR: Escrita com armazenamento na memória não-volátil
- ML_S_WRV: Escreve sem salvar

h +4: Número index do parâmetro a ser modificado ou lido (ver diretório de índice de parâmetros)

h +5: Número da variável h' a qual o dado de leitura é armazenado ou de onde o dado a ser escrito é obtido. Isto significa que é o primeiro número variável da estrutura de dados.
- h' + 0: Contém o dado para os serviços de escrita do parâmetro
- h' +1: Contém o dado que é lido por um serviço de parâmetro
- h' +2: PO1 data da troca de dados de processo
- h' +3: PO2 data da troca de dados de processo
- h' +4: PO3 data da troca de dados de processo
- h' +5: PI1 data da troca de dados de processo
- h' +6: PI2 data da troca de dados de processo
- h' +7: PI3 data da troca de dados de processo

h +6: Contém o código de erro após o serviço ter sido realizado ou contém zero se não houver erro (ver "Return Codes for Parameter Settings" nos manuais "Serial Communication" ou "Fieldbus Unit Profile with List of Parameters").

As estruturas padrão SEW a seguir são disponíveis para o comando _MoviLink:

Comando	Estrutura padrão	Elementos	Descrição breve
_MoviLink	MOVLNK	Bus types	Possíveis tipos de rede: ML_BT_S0: S0 (RS485 #1) ML_BT_S1: S1 (RS485 #2) ML_BT_SBUS: SBus
		Address	Endereço simples (0...99) ou endereço de grupo (100...199) H+1 = 253: Endereço do conversor H+1 = 254: Conexão ponto-a-ponto H+1 = 255: Transmissão Se um endereço de grupo SBus (por ex. 43) é endereçado, o offset 100 deve ser acrescentado. Neste caso, H+1 = 143.
		Format	Especificação dos canais de processo (PD) e de parâmetro (PARAM) para transferência de dados: ML_FT_PAR1: PARAM+1PD ML_FT_1: 1PD ML_FT_PAR2: PARAM+2PD ML_FT_2: 2PD ML_FT_PAR3: PARAM+3PD ML_FT_3: 3PD ML_FT_PAR: Parameter (sem PD)
		Service	Serviço de comunicação ML_S_RD: Serviço de leitura ML_S_WR: Escrita com armazenamento na memória não-volátil ML_S_WRV: Escreve sem salvar
		Index	Número index do parâmetro a ser modificado ou lido (ver diretório index de parâmetro)
		DPointer	Número da variável a qual o dado de leitura é armazenado ou de onde o dado a ser escrito é obtido (estrutura MLDATA)
		Result	Contém o código de erro após o serviço ter sido realizado ou contém zero se não houver erro (ver "Return Codes for Parameter Settings" nos manuais "Serial Communication" ou "Fieldbus Unit Profile with List of Parameters").
	MLDATA	WritePar	Parâmetro que é enviado para serviços de escrita
		ReadPar	Parâmetro que é enviado para serviços de leitura
		PO1	Dados de saída de processo 1
		PO2	Dados de saída de processo 2
		PO3	Dados de saída de processo 3
		PI1	Dados de entrada de processo 1
		PI2	Dados de entrada de processo 2
		PI3	Dados de entrada de processo 3

O elemento DPointer na estrutura MOVLNK deve ser inicializado com o primeiro número variável da estrutura de dados que contém os dados de comunicação.

Se este dado está contido na estrutura com o nome bus data (linha de declaração MLDATA bus data), a linha de inicialização dentro da estrutura MOVLNK deve ler conforme a seguir para o elemento DPointer:

```
***.DPointer = numof(Bus data);
```

MOVLNK é um comando de espera. O próximo comando somente será processado uma vez o comando MOVLNK ter sido completamente realizado.

Caso queira acessar dois ou mais comandos MOVLNK ciclicamente, eles terão que ser processados em **um** task. Este processo é cumprido principalmente para MOVIDRIVE® B no task 2 ou task 3.

Exemplo 1

```
#include <constb.h>

MOVLNK Ml;
MLDATA Mld;

main()
{
  while(1)
  {
    // Initialize structure ml
    ml.BusType  = ML_BT_S1; // RS-485 #2
    ml.Address  = 1;
    ml.Format   = ML_FT_PAR2; // 2 PD with parameter
    ml.Service  = ML_S_RD; // Read
    ml.Index    = 8300; // Index of the inverter status
    ml.DPointer = numof(mld); // Target structure

    _MoviLink( ml ); // Actual command call
  }
}
```

Exemplo 2

Ver a seção –Compilador " Exemplos".

Ajustes de parâmetro para o emissor (mestre)

Endereçamento via RS-485: Nenhum ajuste necessário.

Endereçamento via SBus:

Parâmetros	Endereço	Explicação
P816		A taxa de transmissão depende do comprimento do cabo de rede e deve ser a mesma para o emissor e o receptor.

Ajustes de parâmetro para o receptor

Ampliação de dados via canal de parâmetro

Endereçamento via RS-485 (P810 ... P812)

Parâmetros	Endereço	Explicação
P810	0 ... 99	Endereçamento individual (endereço emissor)
P811	101 ... 199	Endereçamento de grupo (multi-cast), todos os recipientes com o mesmo endereço de grupo podem ser escritos pelo emissor ao mesmo tempo.
P812		Função de monitoração timeout, útil somente com transferência de dados cíclica (desativada se ajustada para 0 ms ou 650 ms)

Endereçamento via SBus P88_ e P89_ para MOVIDRIVE® B / P813 ... P816 para MOVIDRIVE® A

Parâmetros	Endereço	Explicação
P881 / P891 P813	0 ... 63	Endereçamento individual (endereço emissor) (se for utilizada a opção multi-master da SBus, isto é, se vários conversores transmitem o comando MOV-LNK ao mesmo tempo), o serviço de comunicação com o endereço final mais baixo (P813) tem prioridade mais alta
P882 / P892 P814	0 ... 63[1)]	Endereçamento de grupo (multi-cast), todos os recipientes com o mesmo endereço de grupo podem ser escritos pelo emissor ao mesmo tempo.
P883 / P893 P815		Monitoração timeout (desativada com ajuste 0 ms ou 650 ms)
P884 / P894 P816		A taxa de transmissão depende do comprimento do cabo de rede e deve ser a mesma para o emissor e o receptor
P886 ... P888 / P896 ... P898 P817... P819		Não importante na conexão com o comando MOVLNK

1) Quando utilizar o endereço de grupo, aumentar por 100 o valor de entrada para o endereço final.

Os resistores de terminação SBus devem ser ligados ou conectados na primeira e última estações.

Ampliação de dados via canal de dados de processo

A comunicação serial deve ser ajustada de acordo com as tabelas acima (endereçamento via RS-485 / SBus) para a ampliação dos dados de processo. São necessários os seguintes ajustes adicionais para utilizar os dados de processo:

Parâmetros	Explicação
P100	Ajustar a fonte do valor nominal para "RS-485" ou "SBus" (somente se quiser utilizar a especificação do valor nominal através da comunicação de dados de processo)
P101	Ajustar a fonte do sinal de controle para "RS-485" ou "SBus"
P870...876	Descrição dos dados de processo (para uma descrição mais detalhada, consultar o manual "Fieldbus Unit Profile")

É importante ao utilizar o comando MOVLNK que as variáveis (H0...127) e todos os parâmetros salvos permanentemente não estejam escritos ciclicamente na memória volátil com o serviço de comunicação = 2 porque o número de processos salvos com o meio de memória utilizado (EEPROM) é limitado a 10^5 processos salvos.

No MOVIMOT®, a comunicação acíclica desliga a monitoração timeout. Para monitoração de comunicação do timeout cíclico (independente do tempo de funcionamento do programa), o comando _MovComm é disponível em MQx e nós recomendamos utilizar esta opção. Uma vez a comunicação cíclica ter sido iniciada com o comando _MovCommOn, somente é possível o comando _MoviLink para endereço 253 (interno). Ao utilizar o comando _MoviLink, o MOVIMOT® não pode mais ser acessado.

_MovCommDef

O comando _MovCommDef não pode ser utilizado com MOVIDRIVE®.

Sintaxe

```
_MovCommDef(h)
```

Descrição

Os dois comandos MovComm permitem a ampliação dos dados cíclicos entre MQx e normalmente até 4 MOVIMOT® através da interface RS-485 com o protocolo MOVILINK. _MovCommDef é utilizado para configurar a conexão de comunicação com MOVIMOT® pelo ajuste dos parâmetros, como o endereço do conversor. _MovCommOn é utilizado para iniciar a comunicação cíclica. Depois disso, a comunicação cíclica funciona, independente do processamento do comando atual no programa IPOSplus®. É disponível uma cópia dos dados de processo ampliados nas variáveis IPOSplus® e podem ser lidos e escritos lá. A comunicação cíclica pára quando o programa IPOSplus® é parado.

São permitidos até 8 links de comunicação. Observe que o número de links de comunicação tem uma grande influência na duração do ciclo de rede da RS-485 e portanto também no tempo de reação do MOVIMOT®. Devem ser considerados aproximadamente 20 ms da duração do ciclo de rede por link de comunicação ou estação. O pré-requisito para obter a duração do ciclo de rede de 20 ms por estação é o cabeamento livre de irregularidade da RS-485. Se ocorrer um timeout durante a comunicação cíclica, este é indicado na irregularidade 91 Gateway Sysfault. Quando um sinal de retorno é recebido do MOVIMOT®, a mensagem de erro é cancelada automaticamente.

Inserir todas as informações necessárias para executar um comando na estrutura de dados na área da variável com o programa do usuário. O início desta estrutura da variável é o argumento para o comando. A variável é definida pelo **nome MOVCOM;** e tem a seguinte estrutura.

H+0	**Tipo de comunicação (interface)** ML_BT_S1: S1 (RS-485 #2")
H+1	**Endereço individual ou endereço de grupo para o MOVIMOT® a ser endereçado** 0 ... 99 Endereçamento simples 100 ... 199 Endereçamento de grupo 255 Transmissão
H+2	**Entrada dos dados de processo para transferência de dados** 3 = 2 palavras de dados de processo acíclicas (para MOVIMOT®) = ML_FT_2 5 = 3 palavras de dados de processo acíclicas (para MOVIMOT®) = ML_FT_3
H+3	**Número da variável H' a qual os dados de processo são armazenados ou da qual os dados a ser escritos são obtidos**. (A estrutura de dados para H' é descrita abaixo com detalhes.)
H+4	**Número da variável H' a qual os dados de parâmetro são armazenados ou da qual os dados a ser escritos são obtidos**. O MOVIMOT® não realiza esta função.

A estrutura da variável contendo os dados de processo é definida no Compilador pelo **nome da variável MCPDATA;**:

Estrutura de dados para H':	
H'+0	Contém o código de erro após conexão, ou zero se não houver erro 0x05000002 indica que a conexão expirou.
H'+1	PO1 data da troca de dados de processo
H'+2	PI1 data da troca de dados de processo
H'+3	PO2 data da troca de dados de processo
H'+4	PI2 data da troca de dados de processo
H'+5	PO3 data da troca de dados de processo
H'+6	PI3 data da troca de dados de processo

O dado de processo é codificado conforme o MOVILINK.

A estrutura da variável contendo os dados do parâmetro é definida no Compilador pelo **nome da variável MCPARADATA;**:

H+0	Contém o código de erro após o serviço de parâmetro ter sido realizado, ou contém zero se não houver erro. Os erros são codificados conforme MOVILINK.
H+1	0: Nenhuma ação ou ampliação dos dados de parâmetro está completa. 1: Inicia a ampliação dos dados de parâmetro
H+2	ML_S_RD: Serviço de leitura ML_S_WR: Escrita com armazenagem na memória não-volátil ML_S_WRV: Escrita sem salvar
H+3	Número index do parâmetro a ser revisado ou lido
H+4	Dados de leitura após serviço de leitura. Dado a ser escrito no caso de um serviço de escrita.

Ao fazer os ajustes dos parâmetros, proceder conforme a seguir:

1. Entrada do serviço, index e dados
2. Iniciar o processo do ajuste de parâmetros ajustando StartPar para 1.
3. Esperar pelo serviço a ser realizado; o final é indicado quando StartPar é ajustado para 0.
4. Evaluate ParaResult. Se ocorreu um erro, o valor de dados é inválido. Se não ocorreu erro, o serviço foi bem-sucedido.

Argumento

h Primeira variável da estrutura

Exemplo

```
/*============================================
IPOS Source File
============================================*/
#include <const.h>
#include <io.h>

#pragma initials 0 127
#pragma globals 128 300
#pragma var 301 400

MOVCOM mc1;          // control values for communication link to MOVIMOT
MCPDATA mcpd1;         // process data exchange with MOVIMOT
MCPARADATA mcpara;    // parameter data exchange with MOVIMOT (not used)

/*============================================
Main Function (IPOS Entry Function)
============================================*/
main()
{
  // Initialization ============================
  // fill control structure for communication link to MOVIMOT
  mc1.BusType     = ML_BT_S1;     // communication via RS-485 to MOVIMOT
  mc1.Address      = 1;            // MOVIMOT address 1
  mc1.Format     = ML_FT_3;      // PDU type: 3 process data words cyclic
  mc1.PdPointer   = numof(mcpd1); // pointer to process data block
  mc1.ParaPointer = numof(mcpara1); // pointer to parameter data block
  _MovCommDef( mc1 );
  _MovCommOn(  );

  while( )
  {

  }
}
```

_MovCommOn

O comando _MovCommOn não pode ser utilizado com MOVIDRIVE®.

Sintaxe

```
_MovCommOn()
```

Descrição

O comando inicia a comunicação cíclica, os links de comunicação configurados utilizando MovCommDef são ativados. A partir deste ponto, nenhum comando MovCommDef é permitido. Igualmente, nenhum comando MOVILINK para endereço ≠ 253 (interno) pode ser utilizado.

Argumentos

Nenhum

Exemplo

Ver função _MovCommDef

_Nop

Sintaxe

```
_Nop()
```

Descrição

Nenhuma operação é realizada. Este comando pode ser utilizado, por exemplo, para obter tempos de espera na base do comando de duração do ciclo.

Argumento

O comando não tem um argumento.

Exemplo

```
main()
{
  _Nop(  );
}
```

_SBusCommDef

Sintaxe

```
_SBusCommDef( objecttype, h )
```

Descrição

Este comando configura um objeto de dados para transferência cíclica ou acíclica. Este objeto pode ser utilizado para transferir até 2 variáveis (8 bytes) através do system bus. O objeto de dados é escrito para uma estrutura da variável, a variável inicial que é especificada em h. A transferência de dados cíclicos deve ser iniciada com a função SBusCommOn(). O tipo do objeto de dados é especificado em objecttype.

Argumentos

objecttype Expressão que pode adotar um dos seguintes valores:

SCD_TRCYCL: Cyclical send
SCD_REC: Receive
SCD_TRACYCL: Acyclical send

h Primeira variável na estrutura

Podem ser transferidos os seguintes tipos de objeto:

SCD_TRCYCL

Inicializa um objeto de dados, de quem os dados são transmitidos ciclicamente após o comando "SCOMON" ser realizado. O número de objetos que pode ser configurado depende da duração do ciclo.

Duração do ciclo	Número de objetos
1 ... 9 ms	5
10 ... 65530 ms	10

O objeto tem a seguinte estrutura:

h+0: Número do objeto

h+1: Duração do ciclo [ms]

Duração do ciclo válida:

- 1, 2 ... 9 ms
- 10, 20, ... 65530 ms

h+2: Duração do Offset [ms]

Duração do offset válida:

- 0, 1, 2 ... 65534 ms por duração do ciclo < 10 ms
- 0, 10, 20, ... 65530 ms por duração do ciclo >= 10 ms

h+3: Número de bytes de dados e formato de dados

Bit	Valor	Função
0 ... 3	0 ... 8	Número de bytes de dados
4 ... 7	--	Reservado
8	0	Formato MOTOROLA
	1	Formato INTEL
9 ... 31	--	Reservado

h+4: Número da variável H' onde iniciam-se os dados a ser transmitidos

h+5: Resultado do comando SCOM

	0	Capacidade de rede livre em % (valor calculado do conversor)
	-1	Duração do ciclo incorreta
	-2	Configuração de muitos objetos
	-3	Sobrecarga da rede
	Em particular, durante a ampliação de dados entre os escravos, deve-se garantir que o total calculado de utilização da rede não exceda 70 %. A utilização da rede é calculada em bits por segundo utilizando a fórmula: Número de mensagens × Bit/mensagem × 1/duração do ciclo Por exemplo, 2 mensagens com 100 bits em ciclo 1 ms = 200000 bits/s = 200 kBaud Isto resulta na seguinte porcentagem de carga da rede em relação a taxa de transmissão selecionada. Por exemplo, 200 kBaud / 500 kBaud = 40 % < 70 %	

SCD_REC

Inicializa um objeto que recebe os dados. Podem ser configuradas no máximo 32 variáveis.

O objeto tem a seguinte estrutura:

h+0: Número do objeto

h+1: Número de bytes de dados e formato de dados

Bit	Valor	Função
0 ... 3	0 ... 8	Número de bytes de dados
4 ... 7	--	Reservado
8	0	Formato MOTOROLA
	1	Formato INTEL
9 ... 31	--	Reservado

h+2: Número da variável H' do ponto que os dados recebidos são armazenados

SCD_TRACYCL

Inicializa um objeto de dados, de quem os dados são transmitidos imediatamente uma vez.

O objeto tem a seguinte estrutura:

h+0: Número do objeto

h+1: Número de bytes de dados e formato de dados

Bit	Valor	Função
0 ... 3	0 ... 8	Número de bytes de dados
4 ... 7	--	Reservado
8	0	Formato MOTOROLA
	1	Formato INTEL
9 ... 31	--	Reservado

h+2: Número da variável H' onde inicia os dados a ser enviados

h+3: Estado do comando de transmissão

Valor	Descrição
0	Pronto
1	Enviando
2	Envio bem-sucedido
3	Envio de erro

MOVIDRIVE® A: O programa IPOS$^{plus®}$ espera neste comando até a mensagem ter sido enviada. Se nenhuma outra estação estiver conectada, a mensagem não pode ser enviada. O estado de espera somente pode ser finalizado pela função de monitoração, por exemplo, a partir de outro task.

MOVIDRIVE® B: O programa IPOS$^{plus®}$ espera neste comando até a mensagem ter sido enviada, mas por no máximo 10 ms. Os usuários somente podem controlar se o telegrama foi enviado corretamente pela avaliação do estado (h+3) ou pela reação esperada.

Comparação do formato MOTOROLA e INTEL:

	Formato MOTOROLA		Formato INTEL	
CAN byte	0 1 2 3	3 2 1 0	0 1 2 3	0 1 2 3
Variable	H'+1	H'	H'	H'+1
Var. byte	3 2 1 0	3 2 1 0	0 1 2 3	0 1 2 3

As estruturas padrão SEW a seguir são disponíveis para o comando _SBusCommDef:

Comando	Estrutura padrão	Elementos	Descrição breve
_SBusCommDef	SCREC	ObjectNo	Número do objeto
		Format	Número dos bytes de dados e formato de dados
		DPointer	Número da variável de onde o ponto de dados recebidos é armazenado
	SCTRACYCL	ObjectNo	Número do objeto
		Format	Número dos bytes de dados e formato de dados
		DPointer	Número da variável de onde inicia os dados a ser enviados
		Result	Código de retorno para parâmetros de ajuste
		H+3 Estado do comando de transmissão 0 = Pronto 1 = Transmitindo 2 = Transmissão bem sucedida 3 = Erro na transmissão	
	SCTRCYCL	ObjectNo	Número do objeto
		CycleTime	Duração do ciclo [ms] Durações válidas do ciclo: - 1, 2 ... 9 ms - 10, 20, ... 65530 ms
		Offset	Duração de Offset [ms] Durações válidas de offset: - 0, 1, 2 ... 65534 ms para durações do ciclo < 10 ms - 0, 10, 20, ... 65530 ms para durações do ciclo >= 10 ms
		Format	Número dos bytes de dados e formato de dados
		DPointer	Número da variável de onde inicia os dados a ser enviados
		Result	Código de retorno para parâmetros de ajuste

Exemplo

Ver _SBusCommOn

Observe as seguintes regras ao selecionar o número do objeto:

1. Na rede SBus inteira, um número de objeto pode ser configurado somente uma vez para transmissão.
2. Dentro de um conversor, um número de objeto pode ser configurado somente uma vez; ser enviado ou recebido uma vez.

Para os pontos 1 e 2, observe que o firmware reserva seus próprios números de objeto automaticamente:

- O número do objeto inserido nos parâmetros P885 / P895 (P817 para MOVIDRIVE® A) para sincronização SBus.
- Para comunicação através do protocolo MOVILINK, dependendo do endereço SBus no parâmetro P881 / P891 (P813[1)] para MOVIDRIVE® A) e o endereço de grupo SBus no parâmetro P882 / P892 (P814 para MOVIDRIVE® A), são utilizados os seguintes números do objeto:
  - 8 × Endereço SBus + 3 — para dados de saída de processo
  - 8 × Endereço SBus +4 — para dados de entrada de processo
  - 8 × Endereço SBus +5 — para dados de saída de processo síncrono
  - 8 × Endereço SBus +3 + 512 — para serviço de solicitação do parâmetro
  - 8 × Endereço SBus + 4 + 512 — para serviço de reação do parâmetro
  - 8 × Endereço de grupo SBus +6 — para dados de processo de grupo
  - 8 × Endereço de grupo SBus + 6 + 512 — para solicitação do parâmetro de grupo
- Os números do objeto definidos pelo CANopen em DS301 (identificadores) são utilizados para comunicação através do protocolo CANopen.

Para MOVIDRIVE® B ISYNC ou CAM via SBus:

O procedimento de sincronização (sync ID) foi modificado a partir do MOVIDRIVE® A. Ao contrário do MOVIDRIVE® A, para MOVIDRIVE® B você deve garantir que no programa IPOS$^{plus®}$ do conversor mestre, a posição atual seja inicializada **primeiro** e **em seguida,** o objeto sync com _SBusCommDef().

1) Se o MOVIDRIVE® A for conectado através da placa opcional DFC11A para o CANBus, o endereço SBus é tirado do ajuste da chave DIP da DFC11A. Neste caso, o endereço de grupo SBus não está ativo.

_SBusCommOn

Sintaxe

```
_SBusCommOn()
```

Descrição

No MOVIDRIVE® B, o comando foi substituído por _SBusCommState. No entanto, devido a compatibilidade decrescente, ele ainda pode ser utilizado com MOVIDRIVE® B.

Este comando ativa a recepção de dados e a transmissão cíclica dos objetos de dados definidos anteriormente. Os objetos de dados são inicializados utilizando a função SBusCommDef() com o argumento SCD_TRCYCL ou SCD_REC.

Argumento

O comando não tem um argumento.

Exemplo 1

```
#include <constb.h>

#define  DATEN     H20
#declare DATUM1    DATEN:0
#declare DATUM2    DATEN:1
#declare DATUM3    DATEN:2
#declare DATUM4    DATEN:3

#define  INTEL     0x100
#define  ANZ_BYTES 4

SCTRCYCL Obj1;

main()
{
  Obj1.ObjectNo  = 1090;
  Obj1.CycleTime = 10;
  Obj1.Offset    = 0;
  Obj1.Format    = INTEL | ANZ_BYTES // Set high and low bytes
  Obj1.DPointer  = numof(DATEN);
  DATUM1 = DATUM2 = DATUM3 = DATUM4 = 0;

  _SBusCommDef( SCD_TRCYCL,Obj1 ); // Send Obj1 cyclically
  _SBusCommOn(  ); // Start transfer

  while(1)
  {
    // Main program task 1
  }
}
```

Exemplo 2

Ver a seção –Compiler " Exemplos".

_SBusCommState

Sintaxe

```
_SBusCommState ( action )
```

Descrição

Este comando inicia ou pára a recepção de dados e a transmissão cíclica dos objetos de dados que foram pré definidos via SBus 1 ou SBus 2. Os objetos de dados são inicializados utilizando a função SBusCommDef com os argumentos SCD_TRCYCL e SCD_REC.

Este comando somente é disponível no MOVIDRIVE® B.

Argumentos

action pode adotar um dos seguintes valores:

Valor	Argumento	Descrição
0	SCS_STARTALL	Inicia a comunicação cíclica sincronamente a partir do SBus 1 e SBus 2.
1	SCS_STOPALL	Pára a comunicação cíclica sincronamente a partir do SBus 1 e SBus 2.
2	SCS_START1	Inicia a comunicação cíclica a partir do SBus 1.
3	SCS_STOP1	Pára a comunicação cíclica a partir do SBus 1.
4	SCS_START2	Inicia a comunicação cíclica a partir do SBus 2.
5	SCS_STOP2	Pára a comunicação cíclica a partir do SBus 2.

_SetInterrupt

Sintaxe

```
_SetInterrupt( event , function name )
```

Descrição

A função é utilizada para especificação de uma função definida pelo usuário como uma rotina de interrupção. O nome da função é dado como um argumento. Uma interrupção pode ser ativada por vários eventos. O evento desejado é dado como um argumento.

Argumentos

event Expressão constante que pode adotar um dos seguintes valores:

SI_DISABLE: A interrupção é bloqueada
SI_ERROR: Dispara uma interrupção no caso de um erro no sistema
SI_TIMER0: Dispara uma interrupção quando o Timer0 é excedido
SI_TOUCHP1: Inicia uma interrupção no caso de uma mudança de pulso em um terminal touch probe se o touch probe foi ativado

function name Nome da função interrupção. (Importante: Ao contrário de uma chamada de função, somente o nome da função sem () é especificado aqui.)

Exemplo

```
#include <constb.h>

T0Interrupt ()
{
  // Statements of the interrupt routine for timer 0
}

main()
{
  // Inform system of T0 interrupt and start
  _SetInterrupt( SI_TIMER0,T0Interrupt );
  while(1)
  {
    // Main program task 1
  }
}
```

_SetSys

Sintaxe `_SetSys( sys , h )`

Descrição Ajusta o valor de um sistema interno com o valor de uma variável IPOS$^{plus®}$.

Argumentos

h Nome da variável fonte

sys Expressão constante que nomeia o valor do sistema. sys pode adotar um dos seguintes valores:

SS_N11: Valor nominal fixo interno n11
SS_N12: Valor nominal fixo interno n12
SS_N13: Valor nominal fixo interno n13
SS_N21: Valor nominal fixo interno n21
SS_N22: Valor nominal fixo interno n22
SS_N23: Valor nominal fixo interno n23

Importante: O novo valor nominal fixo somente é certo ter sido adotado após 5 ms. Pode-se querer atrasar o processamento do programa após um comando _SetSys por 5 ms com um comando _Wait.

SS_PIDATA: Atualiza os dados PI
– h = Número de ítens dos dados PI
– h + 1 = Dados PE 1
– h +2 = Dados PE 2
– h +3 = Dados PE 3

SS_OPMODE: Ajusta o modo de operação
– h = 11: CFC (controle de rotação)
– h = 12: CFC & torque control
– h = 13: CFC & IPOS (posicionamento)
– h = 14: CFC & synchronous operation (DRS11A)
– h = 16: SERVO (controle de rotação)
– h = 17: SERVO & torque control
– h = 18: SERVO & IPOS (posicionamento)
– h = 19: SERVO & synchronous operation (DRS11A)

SS_IMAX: Ajusta a corrente máxima (somente CFC ou SERVO); Unidade: 0.1 %

SS_POSRAMP: Rampas de posicionamento; unidade: 1 ms
– h = Rampa de posicionamento 1
– h + 1 = Rampa de posicionamento 2

SS_POSSPEED: Rampa de posicionamento; unidade: 0.1 rpm
– h = Rotação Horária
– h + 1 = Rotação Antihorária

SS_OVERRIDE: Liga/desliga o override
– h = 0 -> off
– h = 1 -> on

SS_BREAK: Liga/desliga a função freio
– h = 0 -> off
– h = 1 -> on

SS_RAMPTYPE: Especifica o tipo da rampa de posicionamento (muda P916)
– h = 0 -> linear
– h = 1 -> senoidal
– h = 2 -> quadrática
– h = 3 -> rampa de rede
– h = 4 -> suavização de picos
– h = 5 -> cam disk
– h = 6 -> operação síncrona interna

SS_RESET: Reativa o erro do sistema com o número do erro na variável h
– h = Variável com o número do erro

SS_ACTPOS: Ajusta a posição atual
– h = Valor de posição

SS_SPLINE:
Cálculo interno do conversor de um cam disc analítico. No momento, a função somente é disponível em MCH na versão SD –0C disponível.
O cálculo spline é inicializado através da função do sistema após até 20 pontos da curva (x-y = pares do valor, x = posição mestre, y = posição escravo) terem sido especificados em uma faixa de encoder mestre. O cálculo é então iniciado utilizando `h+0 SplineMode` e um cam disk completo ou um segmento de um cam disk selecionado é preenchido. Atualmente, são disponíveis procedimento spline 0 (para uma operação otimizada) e procedimento spline 1 (para movimentos seção-por-seção e seções retas). O cálculo está completo após ≤ 200 ms.

– h+0 = SplineMode: (Faixa de valor: 0 ... 3)
  - = 0: Interpolação não ativa, ou cálculo é terminado
  - = 1: Inicia a interpolação, inserir os valores interpolados a partir do index 0 iniciando com o came eletrônico (em ordem crescente; isto é, a partir do index 0 até 512).
  - = 2: Inicia a interpolação, inserir os valores interpolados a partir do index 512 iniciando com o came eletrônico (em ordem decrescente L; isto é, a partir do index 512 até 0).
  - = 3: Cálculo do parâmetro preparatório para interpolação concluída; inicia a inserção dos valores interpolados no came eletrônico.
– h+1 = SplineModeControl: reservado
– h+2 = SplineDest: (Faixa de valor: 0 ... 5)
  Número do came eletrônico o qual os valores interpolados são inseridos.
– h+3 = SplineNUser: (Faixa de valor: 2 ... 20)
  Número dos pontos da curva a ser utilizado para interpolação e o processo de cálculo (bit 0 ... bit 4 = número dos pontos da curva, bit 7 = 0: spline 0, bit 7 = 1: spline 1)
– h+4 = SplineX0User: (Somente um valor >= 0 pode ser inserido aqui!)
  Inserir o nr. do ponto da curva do eixo X (mestre).
– h+5 = SplineY0User: (Faixa de valor: long = -2^{31} ... 0 ... (2^{31} –1))
  Valor Y (valor de posição) do 1º ponto da curva; quando ACTPOSSCALE ≠ 0, o valor escalonado deve ser inserido na estrutura
– ...
– h+42 = SplineX19User: (Somente um valor <= 512 pode ser inserido aqui!)
  Inserir o nr. do ponto da curva do eixo X (mestre).
– h+43 = SplineY19User: (Faixa de valor: long = -2^{31} ... 0 ... (2^{31} –1))
  Valor Y do 20º ponto da curva; quando ACTPOSSCALE ≠ 0, o valor escalonado deve ser inserido na estrutura

SS_MULTIAXIS: Cálculo total do conversor de uma trajetória
Disponível somente sob consulta. Ver também o adendo para as instruções de operação 'Special design SK-0C for MCH: Calculated Curves with MCH.“.

As estruturas padrão SEW a seguir, são disponíveis para o comando _SetSys:

Comando	Estrutura padrão	Elementos	Descrição breve
_SetSys	SSPOSRAMP	Up	Rampa de posicionamento de aceleração (ms)
		Down	Rampa de posicionamento de desaceleração (ms)
	SSPOSSPEED	CW	Rotação de posicionamento Horária (0.1 U/min)
		CCW	Rotação de posicionamento Antihorária (0.1 U/min)
	SSPIDATA3	Len	Número de ítens dos dados de entrada de processo a ser enviados
		PI1	Dados de entrada de processo 1
		PI2	Dados de entrada de processo 2
		PI3	Dados de entrada de processo 3
	SSPIDATA10	Len	Número de ítens dos dados de entrada de processo a ser enviados
		PI1	Dados de entrada de processo 1
		PI2	Dados de entrada de processo 2
		PI3	Dados de entrada de processo 3
		PI4	Dados de entrada de processo 4
		PI5	Dados de entrada de processo 5
		PI6	Dados de entrada de processo 6
		PI7	Dados de entrada de processo 7
		PI8	Dados de entrada de processo 8
		PI9	Dados de entrada de processo 9
		PI10	Dados de entrada de processo 10

Exemplo

```
main()
{
  // Set the speed control operating mode
  H0 = 11;
  _SetSys( SS_OPMODE,H0 );
}
```

_SetTask

Sintaxe

```
_SetTask ( control word, function name )
```

Descrição

Esta função é utilizada para determinar uma função definida pelo usuário como um task e para iniciar ou parar este task. O nome da função e a palavra de controle são dados como argumentos.

Argumento

control word Expressão constante que pode adotar um dos seguintes valores:

MOVIDRIVE® B	**MOVIDRIVE® A**
ST2_STOP: Stop task 2	T2_START: Start task 2
ST2_START: Start task 2	T2_STOP: Stop task 2
ST3_STOP: Stop task 3	
ST3_START: Start task 3	

function name Nome da função do task.

Exemplo

```
#include <constb.h>

MyTask3 ()
{
// Statements of task 3
}

main()
{
  // Inform system of task 3 and start
  _SetTask ( ST3_START,MeineTask3 );
  while(1)
  {
    //Main program
  }
}
```

_SetTask2

Sintaxe

```
_SetTask2 ( control word, function name )
```

Descrição

Esta função é utilizada para determinar uma função definida pelo usuário como task 2 e para iniciar ou parar este task. O nome da função e a palavra de controle são dados como argumentos. A palavra de controle e o endereço de partida são ambos ajustados para 0 quando a energia é ligada, isto é o Task2 é desativado.

No MOVIDRIVE® B, o comando foi substituído por _SetTask. No entanto, devido a compatibilidade decrescente, ele ainda pode ser utilizado com MOVIDRIVE® B.

Argumentos

control word Expressão constante que pode adotar um dos seguintes valores:

T2_STOP: Stop task 2
T2_START: Start task 2

function name Nome da função Task2. (Importante: Ao contrário da função call, somente o nome da função sem () é especificado aqui.)

Exemplo

```
#include <constb.h>

MeineTask2 ()
{
  // Statements of task 2
}

main()
{
  // Inform system of task 2 and start
  _SetTask2( T2_START,MeineTask2 );
  while(1)
  {
    //Main program
  }
}
```

_SetVarInterrupt

Sintaxe

```
_SetVarInterrupt ( h1 , function name )
```

Descrição

Este comando não está disponível no MOVIDRIVE® A, somente no MOVIDRIVE® B.

O comando ativa uma interrupção da variável com a estrutura de dados, como da variável h1. Se a condição para a interrupção for realizada, a função 'function name' é executada. O evento para a interrupção é a comparação com um valor da variável (ver H+4). Se a estrutura de dados foi inicializada, durante o tempo de funcionamento, o comportamento da interrupção pode ser adaptado dinamicamente para uma completa VarInterrupt utilizando um comando IPOS$^{plus®}$.

Observe: Os dados da estrutura de dados somente são transferidos quando o comando `_SetVarInterrupt ( h1 , function name )` for acessado (consistência de dados). Uma exceção é a variável `pSourceVar`.

Exemplo: Se o valor da estrutura de dados Hx+3 CompareVar for mudado, por exemplo, o valor somente é levado em consideração com o comando `_SetVarInterrupt ( h1 , function name )`.

Argumentos

h1 Primeira variável de uma estrutura de dados (ver tabela H+0)

function name Nome da função de interrupção. Ao contrário da função call, somente o nome da função sem () é especificado aqui.

Estrutura de dados da interrupção da variável:

Variável	Estrutura do elemento VARINT	Descrição
H+0	Control	0: All VarInterrupt = OFF / Reset 1: Interrupt task 2 2: Interrupt task 3
H+1	IntNum	0 ... 3: Define um número seqüencial da VarInterrupt. Uma interrupção com o número x, que já foi ativada, pode ser reativada durante o tempo de funcionamento do programa com outra estrutura de dados utilizando o comando call `_SetVarInterrupt ( h1 , function name )` quando o mesmo número da interrupção é especificado na nova estrutura de dados na posição H+1. Esta caracterírtica não está disponível para a interrupção do task 1.
H+2	SrcVar	Número da variável de referência cujo valor é comparado com o valor de comparação.
H+3	CompVar	Valor de comparação ou máscara utilizada para comparar o valor da variável de referência H+2.
H+4	mode	0: Sem evento de interrupção. Pode ser utilizado para desativar esta interrupção sem desativar todas. 1: Um dos bits da variável de referência, exibido utilizando a máscara CompVar, mudou seu estado: ([*SrcVar(t) ^ *SrcVar(t-T)] & CompVar) != 0 2:Contanto que o valor da variável de referência seja igual ao valor de comparação (*SrcVar == CompVar) 3:Contanto que o valor da variável de referência não seja igual ao valor de comparação (*SrcVar != CompVar) 4:Contanto que o valor da variável de referência seja maior ou igual ao valor de comparação (*SrcVar >= CompVar) 5:Contanto que o valor da variável de referência seja menor ou igual ao valor de comparação (*SrcVar <= CompVar) 6: O valor da variável de referência e o valor de comparação não são 0 ((*SrcVar & CompVar) != 0) 7: O valor da variável de referência e o valor de comparação são 0 ((*SrcVar & CompVar) == 0) 8: Pulso positivo do bit exibido pelo CompVar 9: Pulso negativo do bit exibido pelo CompVar 10: Quando 2; no entanto, a interrupção somente é processada uma vez sempre que a condição for realizada (pulso ativado) 11: Quando 3; no entanto, a interrupção somente é processada uma vez sempre que a condição for realizada (pulso ativado) 12: Quando 4; no entanto, a interrupção somente é processada uma vez sempre que a condição for realizada (pulso ativado) 13: Quando 5; no entanto, a interrupção somente é processada uma vez sempre que a condição for realizada (pulso ativado)
H+5	Priority	Prioridade da interrupção (1 ... 10); task 2 e task 3 é atribuída a prioridade 0.
H+6	IntEvent	Imagem de processo da variável de referência a partir da *SourceVar para o tempo da interrupção.

Exemplo

Ver "Task Management and Interrupts / Variable interrupts with MOVIDRIVE® B".

_SystemCall

Sintaxe

```
_SystemCall( event , function name )
```

Descrição

A função é utilizada para acessar uma função definida pelo usuário quando ocorre um evento relacionado ao sistema. O nome da função e o evento desejados são especificados como argumentos.

Argumentos

event Expressão constante que especifica quando o nome da função é acessado. Esta expressão pode adotar um dos seguintes valores:

SC_UC:	Incondicional
SC_N0:	Se a rotação for igual a zero
SC_N:	Se a rotação não for igual a zero
SC_NOTPOS:	Se sem posição
SC_TP1:	Se houver mudança de pulso no borne touch probe DI02
SC_NTP1:	Se não houver mudança de pulso no borne touch probe DI02
SC_TP2:	Se houver mudança de pulso no borne touch probe DI03
SC_NTP2:	Se não houver mudança de pulso no borne touch probe DI03

function name Nome da função do evento. (Importante: Ao contrário da função call, somente o nome da função sem () é especificado aqui.)

Exemplo

```
#include <constb.h>

SpeedZero () // Event function
{
// Statements of the event function
}

main()
{
  while(1)
  {
    // Main program task 1
    _SystemCall( SC_N0,SpeedZero );
    // if speed == zero, call function
  }
}
```

_TouchProbe

Sintaxe

```
_TouchProbe( action )
```

Descrição

Permite ou bloqueia uma entrada touch probe. As entradas touch probe são os bornes de entrada DI02 e DI03.

As posições touch probe são armazenadas dentro de 100 µs, independente do processamento do programa existente. O nível do borne deve ter sido alterado por pelo menos 200 µs para ser identificado seguramente. O argumento pode ser utilizado para selecionar a mudança de pulso que causa uma touch probe.

Se ocorrer uma mudança de pulso na entrada permitida, as posições atuais da corrente são salvas nas variáveis especificadas no sistema IPOS$^{plus®}$. Para tomar outra medida, a touch probe deve ser novamente liberada.

As posições da touch probe são armazenadas nas seguintes variáveis:

Encoder	Posição do encoder	Posição da touch probe 1 (DI02)	Posição da touch probe 2 (DI03)
Encoder do motor X15	H511 ActPos_Mot	H507 TpPos1_Mot	H505 TpPos2_Mot
Encoder externo X14	H510 ActPos_Ext	H506 TpPos1_Ext	H504 TpPos2_Ext
Encoder absoluto X62	H509 ActPos_Abs	H503 TpPos1_Abs	H502 TpPos2_Abs
Encoder virtual (somente para MOVIDRIVE® B)	H376	H501 TpPos1_VE	H500 TpPos2_VE

Argumentos

action pode adotar um dos seguintes valores:

TP_EN1:	Permite a entrada touch probe DI02
TP_DIS1:	Inibe a entrada touch probe DI02
TP_EN2:	Permite a entrada touch probe DI03
TP_DIS2:	Inibe a entrada touch probe DI03
TP_EN1_HI:	Permite a entrada touch probe DI02 com aumento de pulso
TP_EN1_LO:	Permite a entrada touch probe DI02 com queda de pulso
TP_EN2_HI:	Permite a entrada touch probe DI03 com aumento de pulso
TP_EN2_LO:	Permite a entrada touch probe DI03 com queda de pulso

Exemplo

```
main()
{
  _TouchProbe( TP_EN1 ); // Enable the touch probe input DI02
}
```

_Wait

Sintaxe

```
_Wait( time )
```

Descrição

Espera pelo período (em milisegundos (ms)) especificado em uma constante.

Argumentos

time Constante que especifica o tempo de espera em milisegundos; nenhuma variável possível.

Se quiser que o tempo de espera seja variável, não utilize o comando WAIT. Em vez disso, inicialize um timer (H487 ... H489) e programe uma rotina até o timer ter ocorrido.

Exemplo

```
Timer_0 = 20000;    // start value 20 s
while( Timer_0 ){}  // wait 20 s
```

_WaitInput

Sintaxe

```
_WaitInput ( level, mask )
```

Descrição

A função espera até um nível específico estar presente nos bornes de entrada. A polaridade desejada do nível de entrada e os bornes relevantes são dados como argumentos. A função espera até todos os bornes de entrada, marcados assim na máscara, terem nível "1" ou nível "0".

Argumentos

level Expressão constante que especifica em qual nível de sinal os bornes devem ser testados. Isto pode adotar um dos seguintes valores:

1:	HIGH level ("1" level)
0:	LOW level ("0" level)

mask Expressão binária constante que especifica os bornes a ser testados. Os bits na expressão têm o seguinte significado

Bit 0:	DI00,	mask = 0b1
Bit 1:	DI01,	mask = 0b10
Bit 2:	DI02,	mask = 0b100
Bit 3:	DI03,	mask = 0b1000
Bit 4:	DI04,	mask = 0b10000
Bit 5:	DI05,	mask = 0b100000
Bit 6:	DI10,	mask = 0b1000000
Bit 7:	DI11,	mask = 0b10000000
Bit 8:	DI12,	mask = 0b100000000
Bit 9:	DI13,	mask = 0b1000000000
Bit 10:	DI14,	mask = 0b10000000000
Bit 11:	DI15,	mask = 0b100000000000
Bit 12:	DI16,	mask = 0b1000000000000
Bit 13:	DI17,	mask = 0b10000000000000
Bit 14-31:	reserved	

Pode ser selecionada uma combinação de entrada ajustando os bits adequados na máscara para 1. Por exemplo, para verificar DI00 e DI03, a máscara deve ser: 0b1001

Exemplo

```
#include <constb.h>

main()
{
  _WaitInput( 1,0b100 );
  // as long as terminal DI02 == HIGH ("1"), wait
}
```

_WaitSystem

Sintaxe

```
_WaitSystem( event )
```

Descrição

A função espera contanto que esteja presente um evento relacionado ao sistema. O evento desejado é dado como um argumento.

Argumentos

event Expressão constante que especifica quando a função espera. Esta expressão pode adotar um dos seguintes valores:

SC_UC:	Incondicional
SC_N0:	Quando a rotação é zero
SC_N:	Quando a rotação não é zero
SC_NOTPOS:	Se sem posição
SC_TP1:	Se houver mudança de pulso no borne touch probe DI02
SC_NTP1:	Se não houver mudança de pulso no borne touch probe DI02
SC_TP2:	Se houver mudança de pulso no borne touch probe DI03
SC_NTP2:	Se não houver mudança de pulso no borne touch probe DI03

Exemplo

```
main()
{
  _WaitSystem( SC_N0 ); // as long as speed == zero, wait
}
```

_WdOff

Sintaxe

```
_WdOff()
```

Descrição

O watchdog está desligado.

Argumento

O comando não tem um argumento.

Exemplo

```
main()
{
  WdOFF();
}
```

_WdOn

Sintaxe

```
_WdOn( time )
```

Descrição

Ajusta o valor do contador do watchdog para o valor especificado em 'time.' Se ocorrer o timer do watchdog, o task 1 e o task 2 são parados e é emitida uma mensagem de erro. A aplicação tem que impedir que o timer do watchdog desacelere ciclicamente reativando o contador. O valor do contador tem que durar pelo menos durante um ciclo do programa principal.

Argumentos

time Valor do contador do watchdog em milisegundos (ms).

Exemplo

```
#define WD_ZEIT 1000

main()
{
  while(1)
  {
    /*
    Perform statements in loop
    The total run time of the statements in the loop
     The loop must not be longer than 1000 ms to prevent
    the watchdog from running down.
    */

    _WdOn( WD_ZEIT ); // Retrigger WD every 1000 ms
  }
}
```

15 Compiler – Exemplos

15.1 Ajuste dos bits e dos bornes de saída

Há duas maneiras de ajustar os bits individuais nas variáveis:

1. A função _BitSet(Hx, y) liga o bit y na variável x para um.
2. A operação OR bit-a-bit Hx | K ajusta aqueles bits na variável x para um, que também são ajustados para um na constante K.

Nos dois casos, a legibilidade do programa pode ser melhorada se a posição do bit ou a constante estiver definida simbolicamente. As funções de ajuste do bit são utilizadas principalmente para ajustar as saídas binárias. Portanto, no exemplo a seguir, a variável H481 (StdOutpIPOS) será utilizada como a variável final da operação. A variável H480 (OptOutpIPOS) seria utilizada adequadamente para endereçar as saídas do opcional. No exemplo, o borne de saída DO02 da unidade básica deve ser ajustado.

Utilizando _BitSet()	Utilizando a operação OR

`#include <const.h>`
`#include <io.h>  // MOVIDRIVE A`
`#include <iob.h> // MOVIDRIVE B`
`main()`
`{`
`  _BitSet( StdOutpIPOS, 2 );`
`}`
	`#include <const.h>`
	`#include <io.h>  // MOVIDRIVE A`
	`#include <iob.h> // MOVIDRIVE B`
	`main()`
	`{`
	`  StdOutpIPOS |= DO02;`
	`}`

O texto fonte pode ser simplificado mesmo se os designadores simbólicos forem utilizados para os comandos correspondentes, assim como as variáveis e constantes:

Using _BitSet():

```
#include <const.h>
#include <io.h>   // MOVIDRIVE A
#include <iob.h> // MOVIDRIVE B
#define SetzeDO02   _BitSet( StdOutpIPOS, DO02);
main()
{
  SetzeDO02
}
```

Se várias saídas forem ajustadas ao mesmo tempo, pode-se acessar a função _BitSet() várias vezes em sucessão ou utilizar a operação lógica OR bit-a-bit. No segundo caso, um comando bastará. Isto reduz a quantidade de código e desta maneira também tem um efeito positivo na operação do programa.

O exemplo a seguir utiliza a operação OR para ajustar DO01 e DO02 ao mesmo tempo.

Utilizando a operação OR:

```
#include <const.h>
#include <io.h>   // MOVIDRIVE A
#include <iob.h> // MOVIDRIVE B
main()
{
  StdOutpIPOS |= DO01 | DO02;
}
```

15.2 Desligamento dos bits e dos bornes de saída

Há duas maneiras de reativar os bits individuais nas variáveis:

1. A função _BitClear(Hx, y) desliga o bit y na variável x.
2. A operação AND bit-a-bit Hx & K ajusta aqueles bits na variável x para zero, que também são ajustados para zero na constante K.

Nos dois casos, a legibilidade do programa pode ser melhorada se a posição do bit ou a constante estiver definida simbolicamente. As funções de desligamento do bit são utilizadas principalmente para reativar as saídas digitais. Portanto, no exemplo a seguir, a variável H481 (StdOutpIPOS) será utilizada como a variável final da operação. A variável H480 (OptOutpIPOS) seria utilizada adequadamente para endereçar as saídas do opcional. No exemplo, o borne de saída DO02 da unidade básica é para ser ajustado para zero.

Utilizando _BitClear()	Utilizando a operação AND

`#include <const.h>`
`#include <io.h>  // MOVIDRIVE A`
`#include <iob.h> // MOVIDRIVE B`
`main()`
`{`
`  _BitClear( StdOutpIPOS, 2 );`
`}`
	`#include <const.h>`
	`#include <io.h>  // MOVIDRIVE A`
	`#include <iob.h> // MOVIDRIVE B`
	`main()`
	`{`
	`  StdOutpIPOS &= ~DO02;`
	`  /*The operator '~' causes bit-by-bit`
	`  negation of DO02. This means all bits`
	`  of DO02 are set to one, except bit 2  */`
	`}`

Se várias saídas forem reativadas ao mesmo tempo, pode-se acessar a função _BitClear() várias vezes em sucessão ou utilizar a operação AND bit-a-bit. No segundo caso, um comando bastará. Isto reduz a quantidade de código e desta maneira também tem um efeito positivo na operação do programa.

O exemplo a seguir utiliza a operação AND para reativar DO01 e DO02 ao mesmo tempo.

Utilizando a operação AND:

```
#include <const.h>
#include <io.h>   // MOVIDRIVE A
#include <iob.h> // MOVIDRIVE B
main()
{
  StdOutpIPOS &= ~DO01 & ~DO02;
}
```

15.3 Verificação dos bits e dos bornes de entrada

Para verificar qual nível um certo borne de entrada tem, deve ser testado um bit dentro de uma variável. A variável H483 (InputLevel), que contém os níveis das entradas digitais, ou a variável h de sua escolha que contém os níveis após a função _GetSys() ter sido realizada.

Testando os bits individuais

Para testar o bit de uma variável, execute a operação AND utilizando uma constante a qual o bit a ser testado é ajustado para um.

Se o resultado é zero, então o bit a ser testado também é zero e desta maneira o nível do borne de entrada é baixo. Se o resultado não é zero, então o bit é um.

O exemplo a seguir ajusta H10 para 1 se a entrada digital DI03 estiver ajustada para um.

Testando o bit individual utilizando H483	Testando os bits individuais ; utilizando _GetSys()

`#include <const.h>`
`#include <io.h>  // MOVIDRIVE A`
`#include <iob.h> // MOVIDRIVE B`
`main()`
`{`
`  if(( InputLevel & DI03 ) != 0)`
`  {`
`    H10 = 1;`
`  }`
`}`
	`#include <const.h>`
	`#define EINGAENGE H1`
	`#include <io.h>  // MOVIDRIVE A`
	`#include <iob.h> // MOVIDRIVE B`
	`main()`
	`{`
	`  _GetSys( EINGAENGE, GS_INPUTS );`
	`  if(( INPUTS & DI03 ) != 0)`
	`  {`
	`    H10 = 1;`
	`  }`
	`}`

Testando vários bits

A fim de testar vários bits de uma variável para um certo estado, exibir os bits a ser testados utilizando a operação AND e compare o resultado com uma constante que corresponde ao modelo do bit a ser testado.

O exemplo a seguir ajusta H10 para 1 se houver um na DI01 e zero na DI03.

```
#include <const.h>
#define DI03   0b1000
#define DI01   0b0010
main()
{
  if(( InputLevel & (DI03 | DI01)) == 0b0010)
  {
    H10 = 1;
  }
}
```

15.4 Verificação de um pulso

Exemplo 1

Além do nível de um borne de entrada, o aumento e a queda de pulso também pode ser verificado e avaliado. Nos exemplos de programas a seguir, a saída DO02 é alternada para DI02 para um pulso positivo ou negativo.

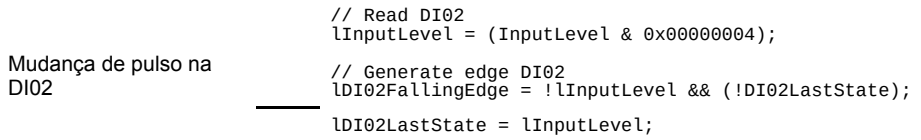
Positive edge query

```
#include <const.h> // MOVIDRIVE A
#include <io.h>    // MOVIDRIVE A

// Variables for edge generation
long  lDI02RisingEdge,
      lDI02LastState,
      lDO02State,
      lInputLevel;

main()
{
   while (1)
  {
    // Read DI02
    lInputLevel = (InputLevel & 0x00000004);

    // Generate edge DI02
    lDI02RisingEdge = lInputLevel && (lDI02LastState);
    lDI02LastState = lInputLevel;

    if(lDI02RisingEdge)
    lDO02State = (!lDO02State)

    // Set output DO02
    if (lDO02State)
      _BitSet( StdOutpIPOS, 2 );

    else
      _BitClear( StdOutpIPOS, 2 );
  }
}
```

Negative edge query

```
#include <const.h>
#include <io.h>

// Variables for edge generation
long  lDI02FallingEdge,
      lDI02LastState,
      lDO02State,
      lInputLevel;

main()
{
   while (1)
  {
    // Read DI02
    lInputLevel = (InputLevel & 0x00000004);

    // Generate edge DI02
    lDI02FallingEdge = !lInputLevel && (lDI02LastState);
    lDI02LastState = lInputLevel;

    if(lDI02FallingEdge)
    lDO02State = (!lDO02State)

    // Set output DO02
    if (lDO02State)
      _BitSet( StdOutpIPOS, 2 );

    else
      _BitClear( StdOutpIPOS, 2 );
    }
}
```

Ao verificar um pulso, observe que a variável auxiliar `lInputLevel` é utilizada para criar o pulso e não o próprio borne de entrada. A variável armazena o estado do borne de entrada antes da operação ser realizada




```
// Read DI02
lInputLevel = (InputLevel & 0x00000004);

// Generate edge DI02
lDI02FallingEdge = !lInputLevel && (!DI02LastState);

lDI02LastState = lInputLevel;
```

Se for utilizado o borne de entrada em vez da variável auxiliar lInputLevel, o pulso no borne de entrada pode mudar no ponto em que o programa IPOS$^{plus®}$ está entre as duas linhas do programa, necessárias para criação do pulso. Isto quer dizer que a mudança do pulso não é identificada no borne de entrada.

Além disso, observe que o pulso de um borne de entrada somente pode ser verificado no task o qual o pulso foi criado. Se não for este o caso, se os tasks individuais não forem síncronos, o pulso não pode ser identificado ou ele pode ser interpretado incorretamente.

Se for necessário o pulso de um borne de entrada em vários tasks, o pulso deve ser criado separadamente em cada task.

Exemplo 2

No exemplo 2, a seção do programa dentro da verificação *if* é processada dependendo do aumento de pulso em DI02.

```
/*============================================
              IPOS source file
============================================*/
#include <constb.h>
#include <iob.h>

long BinInputsNew, BinInputsOld;

/*============================================
   Main function (IPOS initial function)
============================================*/
main()
  {

 /*-------------------------------------
            Initialization
  -------------------------------------*/

 /*-------------------------------------
        Main program loop
  -------------------------------------*/
  while(1)
    {
      // Read binary inputs
      _GetSys( BinInputsNew,GS_INPUTS );
      // Edge query
      if( (BinInputsNew & 0x4) && !(BinInputsOld & 0x4) ) // Rising edge DI02
        {
          // Program statement is located here
        }

      // Save input statuses
      BinInputsOld = BinInputsNew;
    }
  }
```

15.5 Valor absoluto de um número

O exemplo de programa a seguir demonstra como o compilador IPOS$^{plus®}$ pode ser utilizado para criar o valor absoluto de um número.

No task 1 está funcionando um programa de revisão que movimenta o conversor relativamente 400000 incrementos horário e 400000 incrementos antihorário.

No task 2 a rotação atual é lida e a quantidade da rotação atual é criada. A quantidade é armazenada na variável `lActSpeedAbsolute`.

```
#include <constb.h> // MOVIDRIVE A
#include <iob.h>     // MOVIDRIVE B

#define SEKUNDE  1000

SSPOSSPEED tSpeed;
long lActSpeed, lActSpeedAbsolute;

/*==================================
Task 2
==================================*/
Task2()
{
  _GetSys( lActSpeed,GS_ACTSPEED );

  // Calculate absolute speed value
  if( lActSpeed < 0 )
  lActSpeedAbsolute = -lActSpeed;
  else
  lActSpeedAbsolute = lActSpeed;
}

main()
{
  // Initialization =================
  // Positioning speed 500 rpm
  tSpeed.CW = tSpeed.CCW = 5000;
  _SetSys( SS_POSSPEED,tSpeed );

  // Activate Task 2
  _SetTask2( T2_START,Task2 );

  // Main program loop ==============
   while (1)
  {
    _GoRel( GO_WAIT,400000 );
    _Wait( SEKUNDE );
    _GoRel( GO_WAIT,-400000 );
    _Wait( SEKUNDE );
  }
}
```

15.6 Comando MoviLink

O comando `_MoviLink` troca dados ou parâmetros entre as unidades através do SBus ou RS-485. Também é possível ler ou mudar os parâmetros internos.

Os três exemplos a seguir demonstram a função do comando `_MoviLink`:

- Lendo um parâmetro interno: O tipo de referenciamento ajustado é lido utilizando `_MoviLink`.
- Escrevendo uma variável via SBus: Se o MOVIDRIVE® for conectado via SBus, a variável H200 é escrita dependendo do estado da entrada digital DI17.
- Lendo um parâmetro via SBus: A configuração dos dados de processo de um conversor conectado via SBus é lida com o endereço 10.

Lendo um parâmetro interno

```
/*=========================================
Type of reference travel actually entered
in P903 is read in task 1 and written to
variable lRefType.
=========================================*/

/*=========================================
IPOS Source file
=========================================*/

#include <constb.h>
#include <iob.h>

// Definition of MOVLNK structures
MOVLNK tRefType;
MLDATA tData;

// Definition of Variables
long lRefType;

/*=========================================
Main program
=========================================*/

main()
{
  // Initialization of MoviLink for bus transfer
  tRefType.BusType = ML_BT_S1;     // own inverter
  tRefType.Address   = 253           // own inverter
  tRefType.Format = ML_FT_PAR;    // only parameters
  tRefType.Service = ML_S_RD;      // read
  tRefType.Index = 8626;          // P903 RefType
  tRefType.DPointer = numof(tData); // data buffer

  // Main program loop
   while (1)
  {
    // Read type of reference travel
    _MoviLink( tRefType );
    lRefType = tData.ReadPar;
  }
}
```

Escrevendo uma variável via SBus

```
/*=========================================
Variable H200 of inverter connected via
SBus is written depending on DI17 in task 1:
DI17 = 0 -> -1000
DI17 = 1 -> 1000
=========================================*/

/*=========================================
IPOS Source file
=========================================*/

#include <constb.h>
#include <iob.h>

// Definition of MOVLNK structures
MOVLNK tBus;
MLDATA tBusData;

/*=========================================
Main program
=========================================*/

main()
{
  // Initialization of MoviLink for bus transfer
  tBus.BusType = ML_BT_SBUS;      // bus type SBus
  tBus.Address = 10;              // SBus address 10
  tBus.Service = ML_S_WRV;        // write volatile
  tBus.Index = 11200;           // variable H200
  tBus.DPointer = numof(tBusData); // data buffer

  // Main program loop
   while (1)
  {
    if( DI17 )
    {
      tBusData.WritePar = 1000;
      _MoviLink( tBus );
    }
    else
    {
      tBusData.WritePar = -1000;
      _MoviLink( tBus );
    }
  }
}
```

Lendo um parâmetro via SBus

```
/*=======================================
Process data configuration of inverter
connected via SBus is written to variable
lPDData in task 1.
The received values correspond to the
following process data configuration:
0 = PARAM + 1PD
1 = 1PD
2 = PARAM + 2PD
3 = 2PD
4 = PARAM + 3PD
5 = 3PD
6 = PARAM + 6PD
7 = 6PD
8 = PARAM + 10PD
9 = 10PD
=======================================*/

/*=======================================
IPOS Source file
=======================================*/

#include <constb.h>
#include <iob.h>

// Definition of MOVLNK structures
MOVLNK tPD;
MLDATA tData;

// Definition of variables
long lPDData;

/*=======================================
Main program
=======================================*/

main()
{
  // Initialization of MoviLink for bus transfer
  tPD.BusType = ML_BT_SBUS;      // bus type SBus
  tPD.Address = 10;              // SBus address 10
  tPD.Format = ML_FT_PAR;       // only parameters
  tPD.Service = ML_S_RD;         // read
  tPD.Index = 8451;             // P090 PD data config
  tPD.DPointer = numof(tData);    // data buffer

  // Main program loop
   while (1)
  {
    // Read PD configuration
    _MoviLink( tPD );
    lPDData = tData.ReadPar;
  }
}
```

15.7 Comunicação SCOM

O exemplo a seguir mostra um programa que envia duas variáveis ciclicamente a cada 10 ms via SBus. Outro programa recebe o dado que é enviado.

O comando `_SBusCommDef` é utilizado para configurar um objeto de dados para a transferência cíclica de dados. O objeto de transmissão é escrito na estrutura da variável `tBusTr`, o objeto de recebimento é escrito na `TBusRec`.

Para iniciar a transferência cíclica de dados, é acessada a função `_SBusComm0n` para MOVIDRIVE® A e `_SBusCommState` para MOVIDRIVE® B.

Emissor

As variáveis H208 e H209 são transmitidas ciclicamente a cada 10 ms para outro conversor. Os valores da H208 e H209 podem ser mudados utilizando a entrada DI17.

DI17 = 0: H208 = 111111 / H209 = 222222

DI17 = 1: H208 = 222222 / H209 = 444444

O task 2 não é realizado neste exemplo de programa.

Receptor

O receptor obtém os dados de SBus e os armazena nas variáveis H305 e H306.

Receptor

```
/*=========================================
Get data object 1025 from SBus and store
the data in variable H305 and H306.
SHELL settings:
P813 SBus Address -> 2
P816 SBus Baudrate -> 500 kBaud
=========================================*/

/*=========================================
IPOS Source file
=========================================*/

#include <constb.h>
#include <iob.h>

// Definition of SCOM structures
SCREC tBusRec;

// Definition of variables
#define Data_Var1x H305
#define Data_Var2x H306

/*=========================================
Main program
=========================================*/

main()
{
  // Initialization of SCOM transfer object
  tBusRec.ObjectNo = 1060;              // object number
  tBusRec.Format = 8;                  // 8 byte
  tBusRec.Dpointer = numof(Data_Var1x);  // data buffer

  // Start SCOM
  _SBusCommDef( SCD_REC,tBusRec );
  _SBusCommState( SCS_START1 );   // Start cyclic communication MOVIDRIVE B
  // _SBusCommOn( );   // Start cyclic communication MOVIDRIVE A

  // Main program loop
   while (1)
  {
  }
}
```

Emissor

```
/*=======================================
Variables H208 and H209 are sent cyclic
every 10 ms to another inverter via SBus.
The values of H208 and H209 can be altered
with input DI17.
DI17 = 0: H208 = 111111 / H209 = 222222
DI17 = 1: H208 = 222222 / H209 = 444444
SHELL settings:
P813 SBus Address -> 1
P816 SBus Baudrate -> 500 kBaud
=======================================*/

/*=======================================
IPOS Source file
=======================================*/

#include <constb.h>
#include <iob.h>

// Definition of SCOM structures
SCTRCYCL tBusTr;

// Definition of variables
#define Data_Var1 H208
#define Data_Var2 H209

/*=======================================
Main program
=======================================*/

main()
{
  // Initialization of SCOM transfer object
  tBusTr.ObjectNo = 1025;              // object number
  tBusTr.CycleTime = 10;                // cycle time
  tBusTr.Offset = 0;                 // offset
  tBusTr.Format = 8;                 // 8 byte
  tBusTr.DPointer = numof(Data_Var1);  // data buffer
  tBusTr.Result = 1111;              // default value for control

  // Initialize variables
  Data_Var1 = 111111;
  Data_Var2 = 222222;

  // Start SCOM
  _SBusCommDef( SCD_TRCYCL,tBusTr );
  _SBusCommState( SCS_START1 );   // Start cyclic communication MOVIDRIVE B
  // _SBusCommOn( );   // Start cyclic communication MOVIDRIVE A

  // Main program loop
   while (1)
  {
    if( DI17 )
    {
      Data_Var1 = 222222;
      Data_Var2 = 444444;
    }
    else
    {
      Data_Var1 = 111111;
      Data_Var2 = 222222;
    }
  }
}
```

15.8 Processamento de interrupção do touch probe

Os engradados são transportados para uma estação de enchimento em uma esteira temporizada. O sensor de proximidade (DI02) identifica quando o próximo engradado chega. Este dispara uma interrupção e a esteira movimenta-se em uma distância restante especificada. O engradado é então posicionado diretamente sob a estação de enchimento. Uma vez o engradado ter sido enchido, é iniciado um novo ciclo.

A entrada DI10 deve estar ativa durante o ciclo inteiro. Se DI10 = 0, a esteira temporizada pára sujeita ao controle de posição. Quando o conversor é reiniciado (DI10 = 1) ele completa o ciclo.

Os valores de posição, rampa e rotação são inseridos nas variáveis H11 ... H14.

Variável	Nome	Descrição
H11	TP_Max_Pos	Posição final máxima quando a entrada touch probe não é atenuada.
H12	TP_Pos	Distância restante uma vez a entrada touch probe ter sido atenuada.
H13	Speed	Rotação de posicionamento em rpm.
H14	Ramp	Rampa de posicionamento em ms.

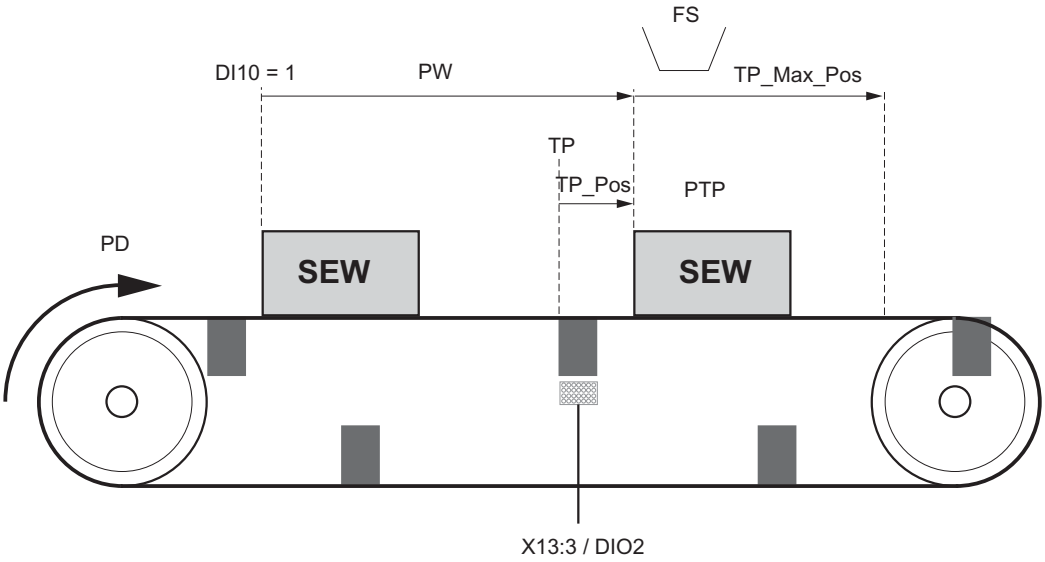


Figura 63: Esteira temporizada com sensor de proximidade

PD: Sentido do pulso
DI10 = 1: Inicia
PW: Largura do pulso
FS: Estação de enchimento
TP_Max_Pos: Posição final máxima
TP: Evento TP
TP_Pos: Distância restante
PTP: Posição após o evento TP
X13:3 / DI02: Sensor de proximidade
SEW: Engradado no transportador de correias

```
/*============================================================================
Description:
A machine cycle is started via input DI10. The target position is the
current motor position (H511) plus TP_May_Pos (H11). If the touch probe input DI02
is not attenuated, the drive moves to this target position. If DI02 is attenuated, a
new target position is calculated. The new target position is calculated from the motor
position
during the touch probe event TpPos1_Mot (H507) plus the remaining distance TP_Pos (H12).

Settings in SHELL :

P601 Binary input DI02  IPOS input
P610 Binary input DI10  IPOS input
P700 Operating mode ... & IPOS
============================================================================*/

#include <const.h>
#include <io.h>

#define CALCTARGET 0
#define BUSSY 1
#define STOP_AKTIV 2

#define State H10
#define TP_Max_Pos H11
#define TP_Pos H12
#define Speed H13
#define Ramp H14

#define h473_ipos_in_position (StatusWord & 0x00080000)   //StatusWord & BIT19

long lPosition;

SSPOSSPEED tPosSpeed;
SSPOSRAMP tPosRamp;
/*============================================
   Interruptroutine Touchprobe
============================================*/
Touchprobe()
  {
    lPosition = TpPos1_Mot + TP_Pos;           //Calculate new target position
    _TouchProbe( TP_DIS1 );                    //Deactivate touchprobe
  }

/*============================================
   Main function (IPOS initial function)
============================================*/
main()
{
  //Initialization
  State = 0;

  // Initialization of the interrupt routine for touch probe input DI02
  _SetInterrupt( SI_TOUCHP1,Touchprobe );

  //Main program loop
  while(1)
  {
    // Set speed and ramp
    tPosSpeed.CW = tPosSpeed.CCW = Speed *10;                // Speed
    tPosRamp.Up = tPosRamp.Down = Ramp;                      // Ramp
    _SetSys(SS_POSRAMP, tPosRamp);
    _SetSys(SS_POSSPEED, tPosSpeed);

    switch(State)
    {
        case CALCTARGET:  if(DI10)
                          {
                            lPosition = ActPos_Mot + TP_Max_Pos;
                            _TouchProbe( TP_EN1_HI ); // Activate rising edge
                            State = BUSSY;
                          }
                          break;

        case BUSSY:       _GoAbs( GO_NOWAIT,lPosition );  // _Go command to variable lPo-
sition

                          if(h473_ipos_in_position && !DI10)
                            // Axis in position and DI10 = 0
                            State = CALCTARGET; // --> Calculate new target position

                          if (!h473_ipos_in_position  && !DI10)
                            // Cancel positioning by setting DI10 = 0
                          {
                            _AxisStop(AS_PSTOP);
                            State = STOP_AKTIV;
                          }
                          break;

        case STOP_AKTIV:  if(DI10) // DI10 = 1 --> Continue positioning
                            State = BUSSY;
                          break;

        default:         break;
    }//switch(State)
  }// while (1)
}// main
```

15.9 Máquina de estado, controle fieldbus com modo de emergência

Um conversor deve ser controlado via fieldbus no modo normal. No caso de uma irregularidade da rede, no entanto, deve ser possível a operação manual utilizando bornes e valor analógico. Além disso, é fornecido o modo misto (setpoint fieldbus + setpoint analógico). O modo de operação é ajustado utilizando bornes de entrada DI10 e DI11. O modo de operação selecionado deve ser indicado nas saídas DO10 e DO11. São fornecidos os seguintes modos de operação:

Modo 0 Controle e setpoint somente via fieldbus

O controle é exclusivamente via fieldbus. Uma palavra de controle reduzida (0 = parada rápida, 6 = liberado) também deve ser utilizada. O setpoint é especificado em termos bipolares via fieldbus (–1500 rpm ... +1500 rpm).

Modo 1 Controle via fieldbus, setpoint = setpoint fieldbus + setpoint analógico

O controle é exclusivamente via fieldbus. Uma palavra de controle reduzida (0 = parada rápida, 6 = liberado) também deve ser utilizada. O setpoint é a soma do setpoint fieldbus (bipolar –1500 rpm ... +1500 rpm) e o setpoint analógico (–10 V ... +10 V = –1500 rpm ... +1500 rpm).

Modo 2 Controle e setpoint via terminal ou valor analógico

O fieldbus é desligado.

Modo 3 Reservado

É realizada uma parada rápida até o modo estar em uso.

Utilizar "IPOS" para a seleção de dados PO e basear a palavra de controle na Control-Word H484, caso contrário haverá um problema no modo 2 se a rede falhar.

O diagrama de estado a seguir mostra as transições entre os modos de operação:

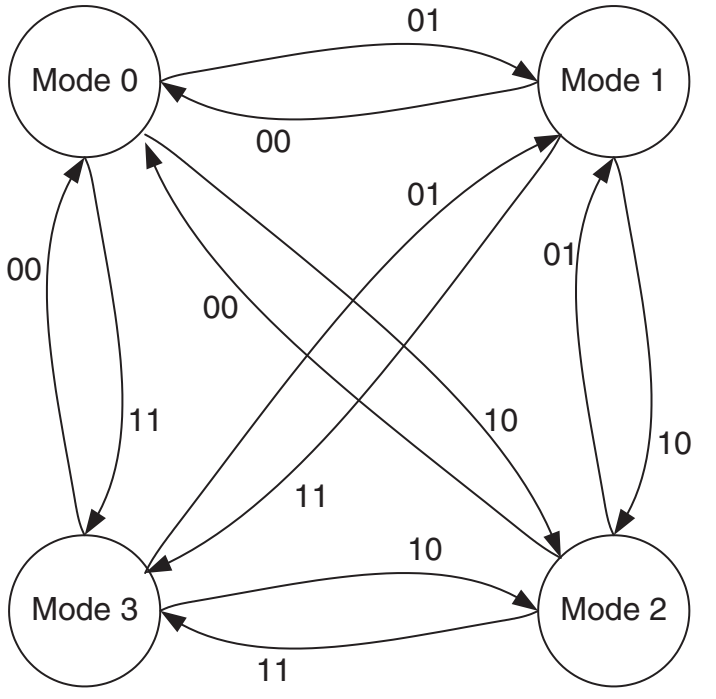


Figura 64: Diagrama dos estados do modo

```
/*=================================================
Operating mode is selected with input terminals
DI10 and DI11 and indicated at the outputs
DO10 and DO11.
The following operating modes are possible:
Mode 0: Control and setpoint via field bus
Mode 1: Control via field bus, setpoint added to
analog value 1
Mode 2: Control via terminals, setpoint analog 1
Mode 3: Reserved

SHELL settings:
P100 Setpoint source: BIPOL./FIX.SETPT
P101 Control signal source: TERMINALS
P600 ... P604 Binary input DI01 ... DI05: NO FUNCTION
P610 / P611 Binary input DI10 / DI11: IPOS INPUT
P630 / P631 Binary output DO10 / DO11: IPOS OUTPUT
P700 Operating mode 1: ... & IPOS
P870 ... P872 Setpoint description PO1 ... PO3: IPOS PO-DATA
P873 ... P875 Actual value description PI1 ... PI3: IPOS PI-DATA
P876 PO data enable: ON
=================================================*/

/*=================================================
IPOS Source file
=================================================*/

#include <constb.h>
#include <iob.h>
#pragma globals 350 399

// Definition of structures
GSPODATA3 busdata; //structure for fieldbus process data
GSAINPUT analog;                    //structure for analog values

// Definition of variables
#define modeselect ((InputLevel >> 6) & 0x00000003)
#define setfixedsetpoint        _SetSys( SS_N11,speed )
#define activatefixedsetpoint _BitSet( ControlWord, 4 )
#define deactivatefixedsetpoint _BitClear( ControlWord, 4 )
#define enable                  _BitClear( ControlWord, 1 )
#define rapidstop               _BitSet( ControlWord, 1 )

// Declaration of variables
long mode, speed, offset;

/*=================================================
Main program
=================================================*/

main()
{
  // Initialization ================================
  // Initialize data structure bus data
  busdata.BusType = 3;          //bus type fieldbus
  busdata.Len = 3;
  busdata.PO1 = 0;
  busdata.PO2 = 0;
  busdata.PO3 = 0;
  // Activate task 2
  _SetTask2( T2_START,buscontrol );

  // Main program loop =============================
   while (1)
  {
  }
}

/*=================================================
Task 2
=================================================*/

buscontrol()
{
  _GetSys( busdata,GS_PODATA );        //get bus data
  mode = modeselect; //read terminals for mode select
  OptOutpIPOS = ((OutputLevel >> 3) & 0xFFFFFFFC) | mode; //output mode
  switch( mode )
  {
    case 0: mode_0();
             break;
    case 1: mode_1();
             break;
    case 2: mode_2();
             break;
    case 3: mode_3();
             break;
  }
}
```

```
/*=================================================
Functions
=================================================*/

mode_0()
{
  if( busdata.PO1 == 6 )
    enable;
  else
    rapidstop;
  speed = busdata.PO2;
  setfixedsetpoint;
  activatefixedsetpoint;
}

mode_1()
{
  if( busdata.PO1 == 6 )
    enable;
  else
    rapidstop;
  _GetSys( analog,GS_ANINPUTS );
  offset = (analog.Input1 + 15) / 10;
  speed = busdata.PO2 + offset;
  setfixedsetpoint;
  activatefixedsetpoint;
}

mode_2()
{
  enable;
  deactivatefixedsetpoint;
}

mode_3()
{
  rapidstop;
}
```

15.10 Estrutura de programação do compilador

O exemplo de programa a seguir pode ser utilizado como a estrutura básica ao criar um programa IPOS$^{plus®}$. Isto inclui uma máquina de estado com quatro modos de operação:

- DISABLE: Nenhum modo de operação é selecionado
- JOGGING: Operação manual
- HOMING: Referenciamento
- POSITIONING: Modo de posicionamento

```
/*======================================================================
Name: Basic_program
Version:    03/07/21 (Y/M/D)
Function:   Basic structure for an IPOS program with state machine
with entry and exit functions for control via
fieldbus or RS485 monitor with 3 I/O process data words


Settings required in SHELL:
---------------------------------------------------------------
P100 = P101 = RS485 for simulation with bus monitor,
= comment FIELDBUS and "#define" for "//constants" for operation with fieldbus ???
P6xx = no functions, exception, e.g. P602 = REFERENCE CAM per reference travel type
P700        = xxx & IPOS
P870 = CONTROL WORD 2
P871        = IPOS PO DATA
P872 = IPOS PO DATA
P873        = STATUS WORD 1
P874        = IPOS PI DATA
P875 = IPOS PI DATA
P916        = e.g. to LINEAR
P960        = OFF (when <>AUS replace: ActPos_Mot => ModActPos and TargetPos => ModTagPos


Drive control via PLC/bus monitor (process output data)
-------------------------------------------------------------------------------------------
---
/DI00 = CONTROLLER INHIBIT

1. word = Control word 2
2. word = jog/positioning speed [1/10 rev./min]
3. word = target position
Assignment of control word 2
15 14 13 12 11 10  9  8  7  6 5  4  3  2  1  0
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Controller inhibit/Enable
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |Enable/rapid stop
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |Enable/stop
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |Hold control
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |Integrator switchover
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |Parameter set switchover
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |Fault reset
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |Start reference travel
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Jog +
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |Jog -
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Operation mode bit 0,  01 = Jog, 10 = Referencing 11 = Automatic
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Operating mode bit 1

Drive feedback to PLC/bus monitor (process input data)
-------------------------------------------------------------------------------------------
---
1. word = status word, user-specific
2. word = actual speed [1/10 rev./min]
3. word = actual position in incr. (only low word)
Assignment of status word, user-specific
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Not assigned
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Inverter ready
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | IPOS reference (drive referenced)
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Target position reached
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Brake released
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Error
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Acknowledge operating mode bit 0,  01 = Jog, 10 = Referencing 11 = Automatic
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Acknowledge operating mode bit 1 (note: AKTIVE BA is acknowledged.)

Brief introduction to using the bus monitor:
1.) Start bus monitor in MOVITOOLS Manager
2.) Click the button Set PO data
3.) In the left-hand field, click on the tab pages PO1, PO2, PO3 and enter the setpoints
4.) Click the "Send" button with the picture of an inverter

-------------------------------------------------------------------------------------------
----
This is sample software, the functionality is NOT guaranteed.
Users accept that in using the sample software they do so at their own risk. SEW does not
guarantee any specific performance.

===========================================================================================
==*/
#include <const.h> //Default path: c:\programme\sew\movitools\projects\include
#include <io.h> //Integrate names of system variables and constants
```

```
//Define ranges of variables
#pragma var H128 H149                // (Default range for compiler aux. variables H400
H419)
#pragma globals H380 H449            // (Default range for global "long" variables H420
H449)
#pragma initials H0 H127             // (Default range for initials H0 H127)
#pragma list // Assembler code with remarks

// Constants
#define MY_PD_LENGTH       3         //3 fieldbus with process data
//#define MY_FBUS_TYPE     GS_BT_FBUS //in GetSys mode to "fieldbus"
#define MY_FBUS_TYPE       GS_BT_S0 //for bus monitor GetSys to "RS485"
#define MY_HALT             13       //Position "Stop" in the IPOS control word H484

// Bit masks
#define MY_OP_MODE           0x18      //virtual inputs DI13/14, InputLevel Bit 9/10
#define MY_READY_TO_RUN      (StatusWord & 0x4) //Ready for operation from H473
#define MY_NO_ERROR          (StatusWord & 0x2) //1 = error-free, 0 = error from H473
#define MY_IN_POSITION       (StatusWord & 0x80000) //IPOS drive has reached target posi-
tion
#define MY_REFERENCED        (StatusWord & 0x100000) //Drive referenced
#define MY_START_HOMING (lPA_ControlWordHigh & 0x1) //Virt. terminal DI10 Start ref.
travel
#define MY_START_POSITIONING (lPA_ControlWordHigh & 0x1) //Virt. terminal DI10 Start posi-
tioning
#define MY_JOG_PLUS          (lPA_ControlWordHigh & 0x2) //Virt. terminal DI11 Jog +
#define MY_JOG_MINUS (lPA_ControlWordHigh & 0x4) //Virt. terminal DI12 Jog -

// Variables for setpoint / actual values, fieldbus control / status word
long lPA_ControlWordHigh; // Bit 8-15 of the fieldbus control word 2
// =DI10 - DI17 of the virtual terminals = bit 6-13 in InputLevel (!!!)
SSPOSSPEED tPosVelocities; //Data structure for positioning speeds
long lPE_StatusWord;      // User status word, bit 8-15 of the fieldbus status word
// =DO10 - DO17 of the virtual terminals = bit 0-7 in OptOutpIPOS (!!!)
long lActPosition,        // Actual position in incr
lScalingNumerator, // Numerator for scaling the position
lScalingDenominator, // Denominator for scaling the position
lActVelocity; // Actual speed in 1/10 rpm

// Variables for controlling operating modes
long lOpMode;              //Operating mode currently selected
long lGlobalStateMachine;  //Status of the global state machine
#define DISABLE       0    //global state machine: Status DISABLE
#define JOGGING 1    //global state machine: Status JOGGING
#define HOMING       2    //global state machine: Status HOMING
#define POSITIONING 3    //global state machine: Status POSITIONING
long lSubStateHoming; //Substatus in main status "Homing"
#define HOMING_STOPPED 0
#define HOMING_STARTED 1
#define HOMING_READY   2
long lSubStatePositioning; //Substatus in main status "Positioning"
#define POSITIONING_STOPPED 0
#define POSITIONING_STARTED 1

//General variables
long lDriveState;         //Inverter status, corresponds to the 7-segment display of MDx
long lErrorCode; //Error code

// Data structures for process data
GSPODATA10 tPA;           //Output data (PLC -> drive)
SSPIDATA10 tPE;           //Input data (drive -> PLC)

/*============================================
Main function (IPOS initial function)
============================================*/
main()
{
  _WdOn( 5000 );  //Activate Watchdog => if there is a fault, code 41
  while (!MY_READY_TO_RUN)
  { //max. 5000 ms wait until the inverter firmware is fully started up
  }
  _WdOff( ); //Deactivate Watchdog startup

  /*-------------------------------------
  Initialization
  -------------------------------------*/

  // Initialize main status
  lGlobalStateMachine = 0;

  // Initialize scaling for the position
  lScalingNumerator = 1;
  lScalingDenominator = 1;

  // Initialize fieldbus variables for Getsys and Setsys commands
  tPA.BusType = MY_FBUS_TYPE; //Process data operation via source s.a.
  tPA.Len = tPE.Len = MY_PD_LENGTH; //PD length s.a.

  // Activate task2
  _SetTask2(T2_START, fnTask2); //To debug task 2, add inverse slashes here and delete
them below
```

```
  /*-------------------------------------
  Main program loop
  -------------------------------------*/
   while (1)
  {
    //Process main state machine
    switch (lGlobalStateMachine)
    {
      // Either no operating mode has been selected, or a selection is not possible
      case DISABLE:
                        break;
      // Operating mode "Jog"
      case JOGGING:     fnJogging();
                        break;
      // Operating mode "Referencing"
      case HOMING:      fnHoming();
                        break;
      //Operating mode"Positioning"
      case POSITIONING: fnPositioning();
                        break;
      //Programming error - invalid status
      default: _AxisStop(AS_PSTOP);
               lGlobalStateMachine = lOpMode = -1;
               break;
    } // End switch (lGlobalStateMachine)
  } // End while(1)
} // End main

/*=============================================
= Function: fnJogMode()
= Jog axis. With 2 inputs, the axis can be moved to the right and to the left.
= If the job mode is not set, the drive remains in hold control. If the jog mode is acti-
vated when the drive is released
= main state machine springs to state 99.
=============================================*/
fnJogging()
{
  // Instructions for entering the main state "Jogging"
  // Acknowledge mode
  _BitSet(lPE_StatusWord,11);
  _BitClear(lPE_StatusWord,12);

  // cyclical processing as long as the main state is set to "Jogging"
  do
  {
    //Read PO data
    _GetSys( tPA.BusType ,GS_PODATA );

    if (MY_JOG_PLUS&&(!MY_JOG_MINUS))
    {
      tPosVelocities.CW = tPosVelocities.CCW = tPA.PO2;
      _SetSys( SS_POSSPEED, tPosVelocities );
      TargetPos = ActPos_Mot + 409600;
    }

    if (MY_JOG_MINUS&&(!MY_JOG_PLUS))
    {
      tPosVelocities.CW = tPosVelocities.CCW = tPA.PO2;
      _SetSys( SS_POSSPEED, tPosVelocities );
      TargetPos = ActPos_Mot -409600;
    }

    if ((MY_JOG_MINUS && MY_JOG_PLUS)||((!MY_JOG_MINUS)&&(!MY_JOG_PLUS)))
      _AxisStop(AS_PSTOP);

  } while (lGlobalStateMachine==JOGGING);

  // Instructions for leaving the main state "Jogging"
  // Stop drive
  _AxisStop(AS_PSTOP);
  // Clear mode
  _BitClear(lPE_StatusWord,11);
  _BitClear(lPE_StatusWord,12);

}// end fnJogging()
```

```
/*============================================
= Function: fnHoming()
= Axis reference travel
= Parameters of the group 97x are effective
= A positive edge on REF-START starts a new reference travel
============================================*/
fnHoming()
{
  // Instructions for entering the main state "Homing"
  // Define substatus
  lSubStateHoming = HOMING_STOPPED;
  // Acknowledge mode
  _BitClear(lPE_StatusWord,11);
  _BitSet(lPE_StatusWord,12);

  // cyclical processing as long as the main state "Homing" is active
  do
  {
    //Read PO data
    _GetSys( tPA.BusType ,GS_PODATA );

    switch (lSubStateHoming)
    {
      case HOMING_STOPPED:   if (MY_START_HOMING)
                             {
                               _Go0(GO0_U_NW_CAM);
                               lSubStateHoming = HOMING_STARTED;
                             }
                             break;
      case HOMING_STARTED:   if (!MY_START_HOMING)
                             {
                               _Go0(GO0_RESET);
                               lSubStateHoming = HOMING_STOPPED;
                             }
                             if (MY_REFERENCED)
                             {
                               lSubStateHoming = HOMING_READY;
                             }
                             break;
      case HOMING_READY:     if (!MY_START_HOMING)
                             {
                               lSubStateHoming = HOMING_STOPPED;
                             }
                             break;
    }
  } while (lGlobalStateMachine==HOMING);
  // Instructions for leaving the main state "Homing"
  // Stop drive
  if (lSubStateHoming==HOMING_STARTED)
  {
    _Go0(GO0_RESET);
    lSubStateHoming = HOMING_STOPPED;
  }
  // Clear mode
  _BitClear(lPE_StatusWord,11);
  _BitClear(lPE_StatusWord,12);

}// End fnHoming
```

```
/*============================================
= Function: fnPositioning()
= Positioning mode
============================================*/
fnPositioning()
{
  // Instructions for entering the main state "Positioning"
  // Define substatus
  lSubStatePositioning = POSITIONING_STOPPED;
  // Acknowledge mode
  _BitSet(lPE_StatusWord,11);
  _BitSet(lPE_StatusWord,12);

  // cyclical processing as long as the main state "Positioning" is active
  do
  {
    //Read PO data
    _GetSys( tPA.BusType ,GS_PODATA );

    switch (lSubStatePositioning)
    {
      case POSITIONING_STOPPED:  if (MY_START_POSITIONING)
                                 {
                                   tPosVelocities.CW = tPosVelocities.CCW = tPA.PO2;
                                   _SetSys( SS_POSSPEED, tPosVelocities );
                                   TargetPos = (lScalingNumerator * tPA.PO3)
                                               / lScalingDenominator;
                                   lSubStatePositioning = POSITIONING_STARTED;
                                 }
                                 break;
      case POSITIONING_STARTED:  if (MY_START_POSITIONING)
                                 {
                                   tPosVelocities.CW = tPosVelocities.CCW = tPA.PO2;
                                   _SetSys( SS_POSSPEED, tPosVelocities );
                                   TargetPos = (lScalingNumerator * tPA.PO3)
                                               / lScalingDenominator;
                                 }
                                 else
                                 {
                                   _AxisStop(AS_PSTOP);
                                   lSubStatePositioning = POSITIONING_STOPPED;
                                 }
                                 break;
    }
  }while (lGlobalStateMachine==POSITIONING);

  // Instructions for leaving the main state "Positioning"
  // Stop drive
  if (lSubStatePositioning==POSITIONING_STARTED)
    _AxisStop(AS_PSTOP);
  // Clear mode
  _BitClear(lPE_StatusWord,11);
  _BitClear(lPE_StatusWord,12);

}// End fnPositioning()
```

```
/*============================================
= Function: fnTask2()
= For time-critical program sections that can run asynchronously
= from task one
=
============================================*/
fnTask2()
{

  // Read inverter status
  _GetSys( lDriveState,GS_SYSSTATE );
  // Read error number
  _GetSys( lErrorCode,GS_ERROR );
  //Read PO data
  _GetSys( tPA.BusType ,GS_PODATA );

  // Create branch distributor/ Select operating mode
  // Virtual fieldbus terminal can only be used if a DIO or DIP is not inserted
  // In this case, use "lPA_ControlWordHigh   = tPA.PI1 >> 8;"
  // otherwise "lPA_ControlWordHigh = InputLevel >> 9;" //Move Bit0 to Bit 0
  lPA_ControlWordHigh = tPA.PO1 >> 8; //Move Bit8 to Bit 0
  lOpMode = (lPA_ControlWordHigh & MY_OP_MODE )>> 3; //Bit 3,4 = Operating mode

  // Create status transitions
  switch (lGlobalStateMachine)
  {
    // Either no operating mode has been selected, or a selection is not possible
    case DISABLE:  if (lDriveState>=0xA)
                   {
                     if (lOpMode==JOGGING)
                       lGlobalStateMachine = JOGGING;
                     if (lOpMode==HOMING)
                       lGlobalStateMachine = HOMING;
                     if (lOpMode==POSITIONING)
                       lGlobalStateMachine = POSITIONING;
                   }
                   break;
    // Operating mode "Jog"
    case JOGGING:  if (lDriveState>=0xA)
                   {
                     if (lOpMode==DISABLE)
                       lGlobalStateMachine = DISABLE;
                     if (lOpMode==HOMING)
                       lGlobalStateMachine = HOMING;
                     if (lOpMode==POSITIONING)
                       lGlobalStateMachine = POSITIONING;
                   }
                   else
                     lGlobalStateMachine = DISABLE;
                   break;
    // Operating mode "Referencing"
    case HOMING:   if (lDriveState>=0xA)
                   {
                     if (lOpMode==JOGGING)
                       lGlobalStateMachine = JOGGING;
                     if (lOpMode==DISABLE)
                       lGlobalStateMachine = DISABLE;
                     if (lOpMode==POSITIONING)
                       lGlobalStateMachine = POSITIONING;
                   }
                   else
                     lGlobalStateMachine = DISABLE;
                   break;
    //Operating mode"Positioning"
    case POSITIONING: if ((lDriveState>=0xA)&&(MY_REFERENCED))
                       {
                         if (lOpMode==JOGGING)
                           lGlobalStateMachine = JOGGING;
                         if (lOpMode==HOMING)
                           lGlobalStateMachine = HOMING;
                         if (lOpMode==DISABLE)
                           lGlobalStateMachine = DISABLE;
                       }
                       else
                         lGlobalStateMachine = DISABLE;
                       break;
    //Programming error - invalid status
    default:           _AxisStop( AS_PSTOP );
                       lGlobalStateMachine = lOpMode = -1;
                       break;
  }// End switch (lGlobalStateMachine)

  //Regenerate process input data and send to PLC
  fnBuildStatusWord(); //Generate status word

  _GetSys(lActVelocity,GS_ACTSPEED); //Read actual speed
  tPE.PI2 = lActVelocity; // Output actual speed

  lActPosition = ActPos_Mot; // Actual position
  tPE.PI3 = (lScalingDenominator * lActPosition) / lScalingNumerator; //Actual position

  _SetSys(SS_PIDATA, tPE.Len); //Send PD
}
```

```
/*============================================
= Function: fnBuildStatusWord()
= Here, bit 0-7 of the status word is generated
= if an error occurs, the other outputs are
= replaced by the error code.
============================================*/
fnBuildStatusWord()
{
  _BitMove(lPE_StatusWord,1, StatusWord,2); //Inverter ready
  _BitMove(lPE_StatusWord,2, StatusWord,20); //IPOS referenced
  _BitMove(lPE_StatusWord,3, StatusWord,19); //Target position reached
  _BitMoveNeg(lPE_StatusWord,4, StatusWord,1); //Error
  if ( !MY_NO_ERROR )
  { // if an error occurs, overwrite the status bits of the operating modes with the
    // error code
    lPE_StatusWord = lPE_StatusWord & (lErrorCode << 8);
  }
  tPE.PI1 = lPE_StatusWord ;
}//end fnBuildStatusWord()
```

16 Mensagens de erro

16.1 Mensagens de erro do compilador

Os erros do texto fonte que são reconhecidos pelo pré-processador e pelo compilador são divididos em classes de irregularidade e códigos da irregularidade.

Classe de irregul.	Código da irregul.	Causa possível
STATEMENT	NOT FOUND SEMICOLON	Comandos faltantes do corpo da rotina Ponto e vírgula faltante após comando
CONDITIONAL	COLON	Dois pontos faltantes do comando condicional ":"
BLOCK	END	Bloco sem fechamento dos colchetes " }"
BREAK	SEMICOLON	Ponto e vírgula";" faltante após quebra
CASE	ILLEGAL TYPE COLON DEFAULT	O Case deve ser seguido pela constante O Case constante deve ser seguido por dois pontos A derivação padrão contém erro(s) ou está na posição errada
COMPILER	Error text	Erro interno do sistema (consultar a SEW)
CONTINUE	SEMICOLON	Ponto e vírgula ";" faltante após o *continue*
DECLARE	IDENTIFIER NO VARIABLE TOO MANY #DEFINE	O identificador após #declare é inválido O #declare deve descrever uma variável O número do #define excede os recursos
DEFINE	IDENTIFIER SYMBOL SEQUENCE TOO MANY #define	O identificador após #define é inválido A seqüência do símbolo após #define é inválida O número do #define excede os recursos
DO	WHILE OPEN BRACKET CLOSE BRACKET SEMICOLON	*while* está faltando após o comando *do* Abertura do parênteses "(" faltante após *while* Fechamento do parênteses ")" faltante após *while* Ponto e vírgula ";" faltante após *while*
FACTOR	CLOSE BRACKET	Fechamento do parênteses ")" faltante após expressão entre parênteses
FCT.CALL	CLOSE BRACKET NUMBER ARGS	Fechamento do parênteses ")" faltante após o nome da função O número de argumentos está incorreto
FOR	OPEN BRACKET SEMICOLON CLOSE BRACKET	Abertura do parênteses "(" faltante após *for* Ponto e vírgula ";" faltante entre as expressões *for* Fechamento do parênteses ")" faltante após *for*
FUNCTION	OPEN BRACKET CLOSE BRACKET	Parêntes(es) faltante para declaração da função.
IDENTIFIER	NOT FOUND	Identificador desconhecido
IF	OPEN BRACKET CLOSE BRACKET	Abertura do parênteses "(" faltante após *if* Fechamento do parênteses ")" faltante após *if*
CONSTANT	ILLEGAL TYPE	A sintaxe da constante dec., binária ou hex está incorreta
PRAGMA	IDENTIFIER VARIABLE RANGE	Palavra-chave inválida após #pragma Não é permitida faixa variável
PREPROCESSOR	NO VARIABLE TOO MANY #include SOURCE TEXT TOO LONG HEADER FILE NAME OPEN FILE CLOSE FILE LINES TOO LONG	O nome da variável deve seguir numof Muitas diretivas #include aninhadas O texto fonte excede o comprimento máximo permitido Nome inválido do arquivo de cabeçalho O arquivo não pode ser aberto Final do arquivo inesperado alcançado Linha do texto fonte muito comprida
RETURN	SEMICOLON	Ponto e vírgula ";" faltante após *return*
SWITCH	OPEN BRACKET CLOSE BRACKET	Nenhuma abertura de parênteses "(" após contato ou abertura de chaves "{" faltante do bloco Fechamento do parênteses ")" faltante após contato e fechamento de chaves "}" faltante após bloco
UNDEF	IDENTIFIER	O identificador após #undef é inválido
WHILE	OPEN BRACKET CLOSE BRACKET	Abertura do parênteses "(" faltante após *while* Fechamento do parênteses ")" faltante após *while*

17 Assembler – Introdução

17.1 Características técnicas

Resolução do encoder	O IPOS$^{plus®}$ sempre funciona com 4096 incrementos / volta do motor (pré-requisito: resolução do encoder de 128, 256, 512, 1024 ou 2048 pulsos / volta do motor (não são válidas outras resoluções do encoder) ou resolver)	
Comprimento máx. do programa	800 linhas do programa para MOVIDRIVE® A 1600 linhas do programa para MOVIDRIVE® B (máx. 1000 linhas de programa e de comentário no editor)	
Duração de processamento do comando	MOVIDRIVE® A: Task 1: 1.0 ms / linha do programa Task 2: 0.5 ms / linha do programa MOVIDRIVE® B: Pode ser ajustado no parâmetro P838 / P839, ver "Task Management and Interrupts / Tasks for MOVIDRIVE® B"	
Variáveis	MOVIDRIVE® A: 512, o qual 128 (0 ... 127) pode ser armazenado na memória não-volátil MOVIDRIVE® B: 1023, o qual 128 (0 ... 127) pode ser armazenado na memória não-volátil Faixa de valor: -2^{31} ... $+(2^{31}-1)$	
Entradas touch probe	2 entradas, tempo de processamento 200 µs	
Intervalo de amostragem das entradas digital e analógica	1 ... 5 ms	
Entradas/saídas digitais	MOVIDRIVE® A: MOVIDRIVE® B: Opcional DIO11A: Opcional DIP11A:	6 entradas / 3 saídas 8 entradas / 5 saídas 8 entradas / 8 saídas 8 entradas / 8 saídas
Entradas/saídas analógicas	Unidade básica: Opcional DIO11A:	1 entrada (0 ... 10 V, ± 10 V, 0 ... 20 mA, 4 ... 20 mA) 1 entrada (0 ... 10 V, ± 10 V, 0 ... 20 mA) 2 saídas (±10 V, 0 ... 20 mA, 4 ... 20 mA)

17.2 Ajuste das unidades de deslocamento do usuário

No cabeçalho do programa do assembler, os fatores da distância de deslocamento "NUMERATOR, DENOMINATOR" e "UNIT" podem ser inseridos para determinar a unidade de deslocamento do usuário (por ex. mm, rev., ft).

Fatores da distância de deslocamento NUMERATOR/ DENOMINATOR

O IPOS$^{plus®}$ sempre funciona com 4096 incrementos / volta do motor. O usuário pode querer programar comandos de deslocamento nas unidades do usuário exceto incrementos/volta do motor (por ex. mm, revs., ft.). Neste caso, os fatores da distância de deslocamento "NUMERATOR" e "DENOMINATOR" devem ser ajustados como descrito abaixo. As exceções são os comandos de deslocamento com variáveis como seus argumentos, que podem ser especificados somente em incrementos/volta do motor.

A conversão é definida pela seguinte fórmula:

$$\text{Incrementos} = \frac{\text{NUMERADOR}}{\text{DENOMINADOR}} \cdot \text{unidade de deslocamento do usuário}$$

Fator da distância de deslocamento NUMERATOR

Número de incrementos que o motor movimenta para percorrer uma distância definida.

Faixa de ajuste: 0 ..... **1** ..... $2^{31} – 1$

Fator da distância de deslocamento DENOMINATOR

A distância definida expressa nas unidades de deslocamento do usuário.

Faixa de ajuste: 0 ..... **1** ..... $2^{31} – 1$

Exemplo

12376 incrementos correspondem a 120 mm. NUMERATOR = 12376, DENOMINATOR = 120, UNIT = mm.

Se o numerador ou o denominador são valores não-inteiros, a conversão pode ser feita com mais precisão, se ambos forem multiplicados pelo mesmo fator de expansão (por ex. 10, 100, 1000, etc.). Esta expansão não limita a faixa máxima de deslocamento.

Exemplo

Os três exemplos a seguir demonstrarão como os fatores de deslocamento numerador / denominador de uma unidade linear são ajustados para especificação de posição:

- Exemplo A: Especificação de posição em mm para o eixo linear
- Exemplo B: Especificação de posição em incrementos
- Exemplo C: Especificação de posição em voltas na saída

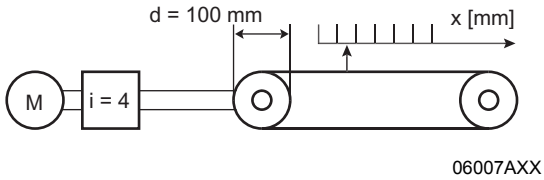


Figura 65: Estrutura mecânica da unidade linear

*Exemplo **A**: mm*

A distância definida a ser calculada é uma volta da engrenagem acionada.

- Fator da distância de deslocamento NUMERATOR = Incrementos/volta do motor × redução i = 4096 × 4 = 16384
- Fator da distância de deslocamento DENOMINATOR = Diâmetro de saída × π = 314.15926

O fator da distância de deslocamento DENOMINATOR não é um número inteiro, deste modo a precisão da conversão pode ser aumentada utilizando um fator de expansão. O fator de expansão deve ser o maior possível, embora o resultado não deve exceder a faixa de ajuste (fator de expansão neste exemplo: 100,000).

- Fator da distância de deslocamento NUMERATOR = 16384 × 100000 = 1638400000
- Fator da distância de deslocamento DENOMINATOR = 314.15926 × 100000 = 31415926

Já que π não é um número finito, a especificação da posição final sempre conterá erros

*Exemplo **B**: Incrementos*

- Fator da distância de deslocamento NUMERATOR = 1
- Fator da distância de deslocamento DENOMINATOR = 1

*Exemplo **C**: Voltas na saída*

- Fator da distância de deslocamento NUMERATOR = Incrementos/volta do motor × redução i = 4096 × 4 = 16384
- Fator da distância de deslocamento DENOMINATOR = 1

Informação prática

Informação prática para determinar o fator da distância de deslocamento durante a colocação em operação.

Por exemplo, ajuste das unidades de deslocamento do usuário em mm.

1. Ajustar os fatores da distância de deslocamento NUMERATOR e DENOMINATOR para o valor 1 (→ unidades de deslocamento do usuário = incrementos).
2. No modo manual, movimentar qualquer número das unidades de deslocamento do usuário (neste caso, incrementos), por ex. 100 000 incrementos.
3. Medir a distância abrangida no sistema no ponto 2, por ex.:
   - Posição inicial = 1000 mm
   - Posição final = 1453 mm
   - Distância abrangida = 453 mm
4. Inserir os fatores da distância de deslocamento no cabeçalho do programa do Assembler:
   - Fator da distância de deslocamento NUMERATOR= 100 000
   - Fator da distância de deslocamento DENOMINATOR= 453

UNIT

Todas as entradas da distância de deslocamento são indicadas com uma unidade na janela do programa.

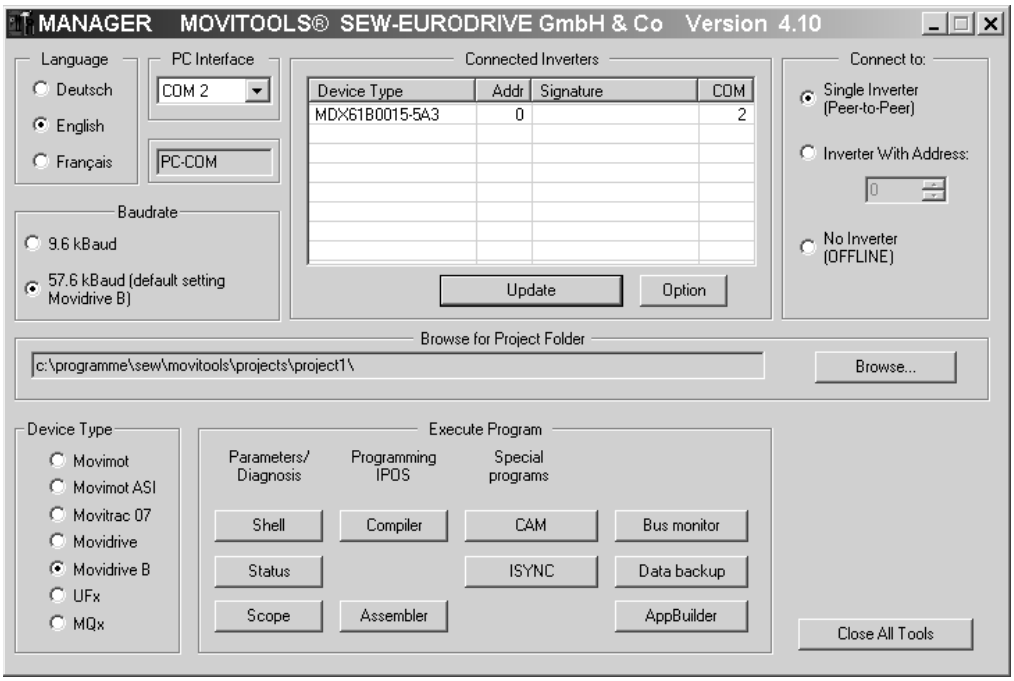
Esta unidade pode ser inserida no cabeçalho do programa para UNIT e pode ter até cinco caracteres no comprimento.

Esta entrada é puramente simbólica e não tem qualquer efeito na função do conversor

17.3 Primeira fase

Iniciar IPOS$^{plus®}$ assembler

O IPOS$^{plus®}$ assembler é iniciado a partir do MOVITOOLS® Manager. O MOVITOOLS® Manager está localizado no menu de colocação em operação do Windows em Programs / MOVITOOLS x.xx / MT Manager.



10453AEN

Figura 66: MOVITOOLS® Manager

Conectar o MOVIDRIVE® via USS11A (MOVIDRIVE® A) ou UWS21A (MOVIDRIVE® B) para uma porta serial livre no PC utilizando um cabo nulo serial do modem. Selecionar esta porta no grupo interface PC (PC-COM). Na figura acima, COM 2 é a porta selecionada.

O MOVIDRIVE® deve ser conectado à rede ou à uma fonte externa 24 V_{CC} para permitir a comunicação da unidade via porta serial.

Clicando na tecla [Update] faz com que o programa procure todas as unidades conectadas e as indique na lista acima delas. A unidade agora deve estar indicada na lista. Se não estiver, não há conexão entre o PC e o conversor. Favor verificar a conexão.

Escolhendo uma unidade desta lista você confirma a escolha da unidade e comuta-a para o modo online.

A interface do programa a seguir é indicada quando iniciar o IPOS$^{plus®}$ assembler:

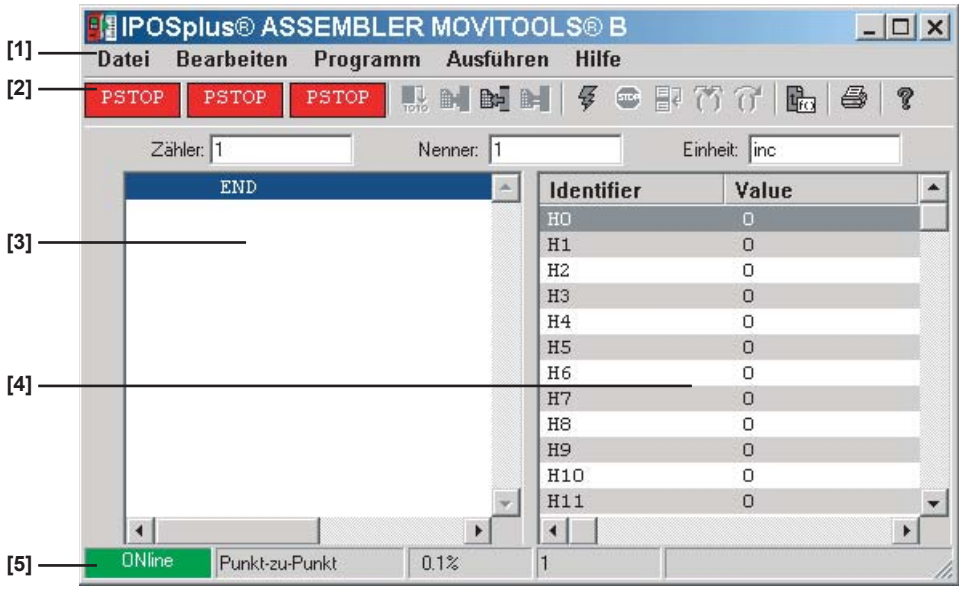
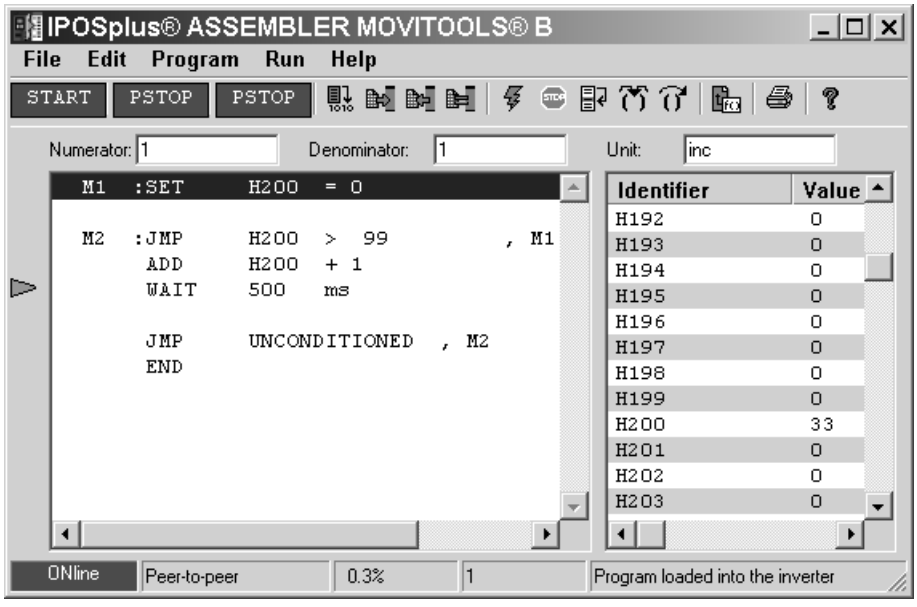


Figura 67: Interface Assembler

[1] Barra do menu
[2] Barra de ferramenta
[3] Janela do programa
[4] Janela de variáveis
[5] Barra de estado

Criando um novo programa

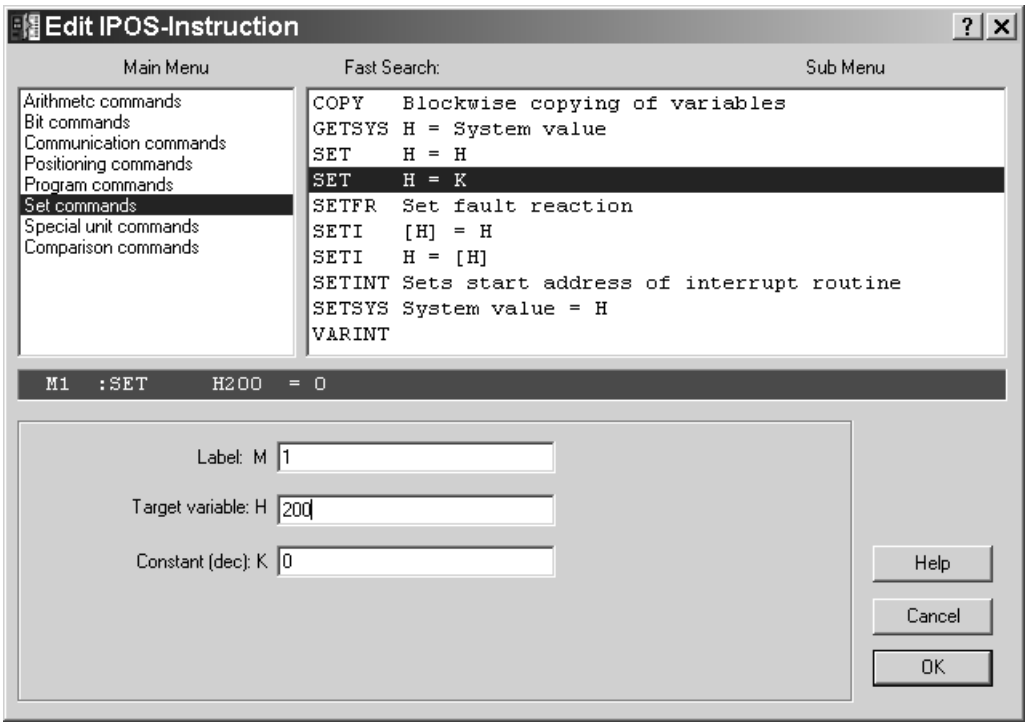
Para familiarizar-se com o IPOS$^{plus®}$ assembler, escrever seu primeiro programa para incrementar uma variável de 0 até 99 em etapas de 500 ms.



10532AEN

Figura 68: Incrementação das variáveis

Os comandos assembler são inseridos utilizando a ferramenta de inserção. Para abrir a caixa de diálogo da ferramenta de inserção, clicar no ícone .



10533AEN

Figura 69: Ferramenta de inserção

Inserir o primeiro comando assembler no programa, na janela [main menu], clicar em [Set commands] e escolher "SET H = K" à direita da janela.

Na seção inferior da caixa de diálogo, inserir a label de salto da linha de comando, a variável final e o valor (constante) o qual a variável deve ser ajustada. Clicar em [OK] para fechar a ferramenta de inserção e inserir o comando no programa.

Utilizar agora a ferramenta de inserção para inserir os comandos restantes no programa. A tabela a seguir indica os parâmetros para todos os comandos do programa. Caso necessite informação em um comando, destaque-o e pressione a tecla <F1>.

Comando	Label	Alvo	Condição	Constante	Destino
SET	1	200		0	
JMP	2	200	>	99	1
ADD		200		1	
WAIT				500	
JMP			UNCONDI-TIONED		2

Compilando e iniciando o programa

Para gerar um programa Assembler de forma que o conversor possa entender, o código fonte deve ser compilado.

Para isto, escolher [Program] / [Compile] ou clicar no ícone na barra de ferramenta.

Se o programa for compilado com sucesso, esta informação é indicada na barra de estado.

Na etapa seguinte, o programa compilado deve ser carregado no conversor. Para isto, escolher [Program] / [Compile + download] ou clicar no ícone na barra de ferramentas.

A barra de estado mostra se foi feito o download do programa com sucesso.

O programa IPOS$^{plus®}$ agora é armazenado na memória não-volátil da unidade.

Também pode ser feito o download dos programas IPOS$^{plus®}$ de um MOVIDRIVE® para outro MOVIDRIVE® utilizando um controle manual DBG60B. Isto é feito utilizando os parâmetros P807 (Copia MDX -> DBG) e P806 (Copia DBG -> MDX).

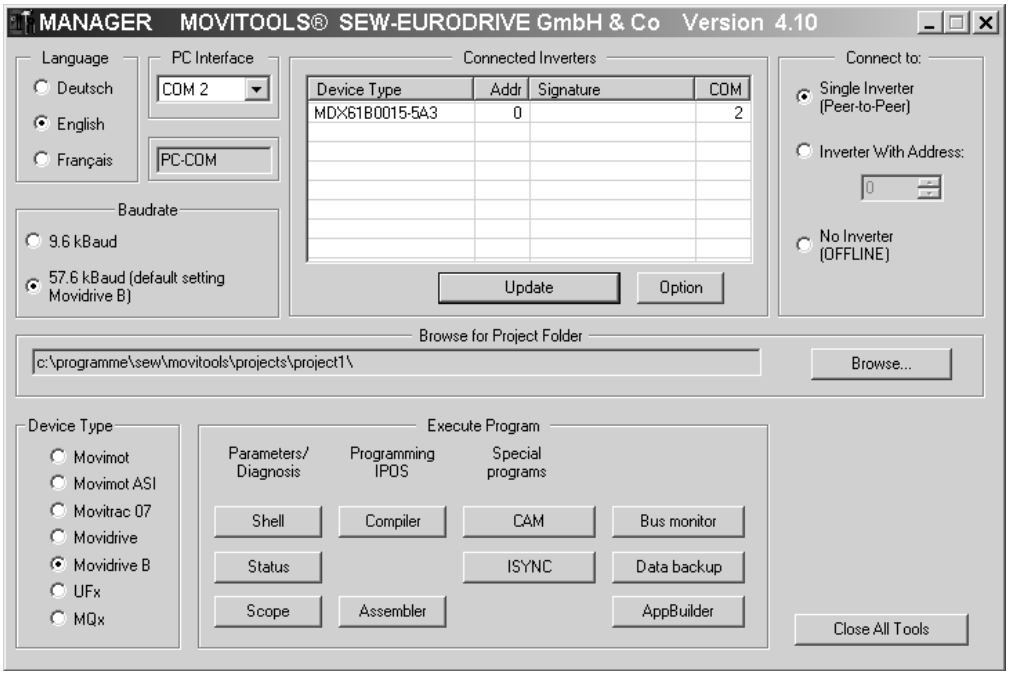
Um programa pode ser iniciado depois de ter sido feito o download para o conversor. Para isto, escolher [Run] / [Start]. Também, pode-se clicar no ícone na barra de ferramentas. Uma vez o programa ter sido iniciado, é indicado uma seta verde (indicador do programa) na janela de projeto para destacar a linha do programa que está sendo atualmente processada. O display na barra de ferramentas muda de PSTOP para START.

Para parar os programas no task 1, task 2 e task 3, escolher [Run] / [Stop] da barra de menu. Também, pode-se clicar no ícone na barra de ferramentas. Depois do programa ter sido parado (todos os tasks), o indicador do programa muda para vermelho e permanece na primeira linha de comando do task 1. A indicação de estado para a seqüência do programa na barra de ferramentas muda de START para PSTOP.

18 Assembler – Editor

O IPOS$^{plus®}$ Assembler é parte do pacote do programa MOVITOOLS®. Pode-se iniciar o IPOS$^{plus®}$ Assembler a partir do MOVITOOLS® Manager.

Antes de iniciar o IPOS$^{plus®}$ Assembler, escolher um diretório de trabalho clicando na tecla [Browse]. Este diretório de trabalho é utilizado quando o dado é carregado e salvo.



10453AEN

Figura 70: MOVITOOLS® Manager

A tela a seguir aparece após o IPOS$^{plus®}$ Assembler ter sido iniciado:

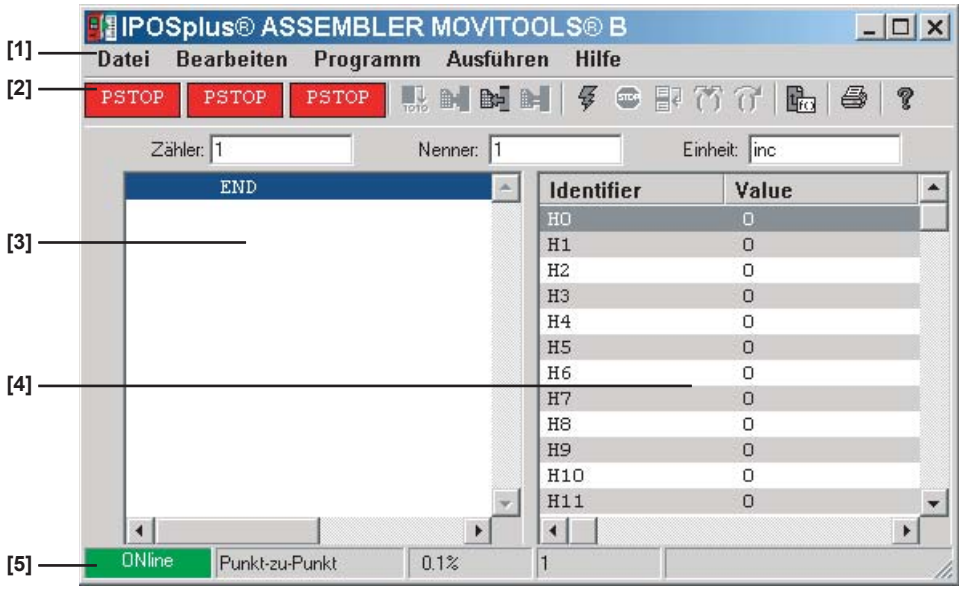


Figura 71: Interface Assembler para MOVIDRIVE® B

[1] Barra do menu
[2] Barra de ferramenta
[3] Janela do programa
[4] Janela de variáveis
[5] Barra de estado

A barra de estado mostra se o equipamento está online ou offline. Ela também indica o conteúdo da memória do programa como porcentagem e o número de linhas do programa selecionado.

Há três campos de entrada sob a barra de ferramentas:

- Numerator
- Denominator
- Unit

Os valores de ajuste de posição podem ser escalonados utilizando a relação numerador/denominador, que quer dizer que eles podem ser especificados nas unidades definidas pelo usuário. As posições especificadas através das variáveis não podem ser escalonadas utilizando esta relação.

Exemplo

O encoder do motor fornece 4096 incrementos por volta. Há um fuso no motor com passo de 10 mm/volta que movimenta um transportador horizontalmente.

O usuário quer especificar as posições que o conversor se movimenta em mm.

Neste caso, ajustar o numerador e o denominador conforme a seguir.

- Numerator: 4096
- Denominator: 10
- Unit: mm

Depois de inserir um comando de posicionamento, pode-se inserir a posição desejada em mm, contanto que o valor seja uma constante.

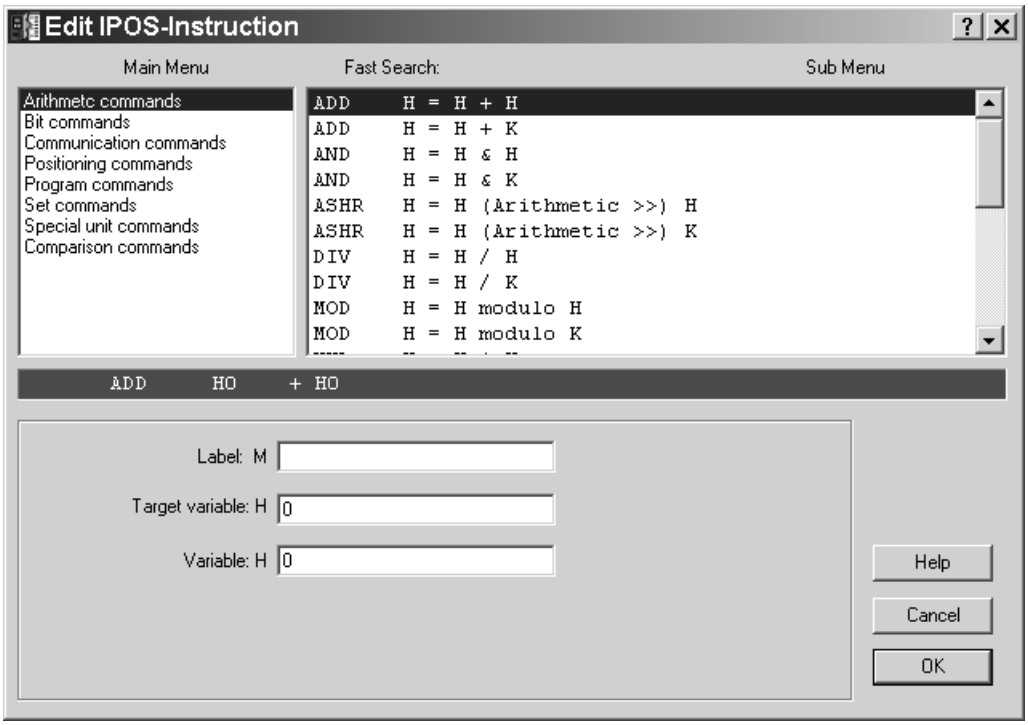
Os estados de operação dos programas do usuário task 1, task 2 e task 3 são:

- START (o programa está em funcionamento)
- PSTOP (programa parado);
- BREAK (o programa somente é processado até a linha marcada)
- STEP (o programa é processado linha-a-linha pressionando a tecla F7).

18.1 Criando programas

Inserindo linhas de comando

Abrir a ferramenta de inserção clicando no ícone , pressionando a tecla <Einfg> ou escolhendo [Edit] / [Insert instruction...] da barra de menu.



10536AEN

Figura 72: Ferramenta de inserção

Todos os comandos disponíveis no IPOS$^{plus®}$ podem ser selecionados na ferramenta de inserção.

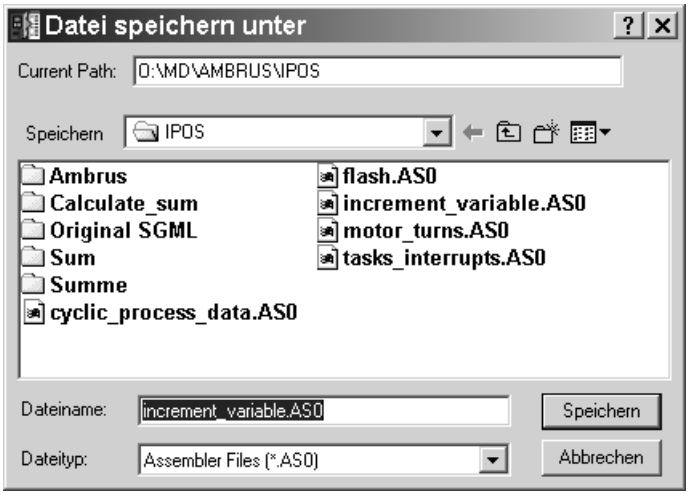
Quando selecionar um comando, aparece uma tela de entrada a qual você tem que inserir os argumentos disponíveis para a função selecionada. Depois de ter inserido todos os argumentos, pressionar [OK] para inserir o comando no programa.

Pode-se utilizar a tecla <Delete> para remover as linhas de comando selecionadas do programa.

Do mesmo modo, pode-se inserir blocos de comando inteiros destacando a seção desejada com o mouse e escolhendo [Edit] / [Copy] e [Edit] / [Insert] ou apagando-os escolhendo [Edit] / [Delete].

Pode-se mudar um comando inserido, com um clique duplo na linha de comando na janela do projeto ou escolhendo [Edit] / [Edit instruction...] da barra de menu.

Salvar o programa Assembler compilado escolhendo [File] / [Save...].



10537AEN

Figura 73: Salva um programa

No MOVIDRIVE® A, os programas Assembler são salvos com a extensão *.MDX. No MOVIDRIVE® B eles são salvos com a extensão *.AS0. Na caixa de diálogo, inserir o nome e o diretório do programa Assembler.

Os arquivos MDX criados utilizando o Assembler contém somente código do programa e nenhum parâmetro.

Os arquivos MDX, criados para armazenar parâmetros utilizando o SHELL, contém parâmetros e código do programa.

Portanto, você deve ter cuidado quando sobrescrever arquivos MDX existentes.

18.2 Compilando e efetuando download

Para gerar um programa Assembler de forma que o conversor possa entender, o código fonte deve ser compilado.

Para isto, escolher [Program] / [Compile + download] da barra de menu ou clicar no ícone na barra de ferramentas.

Se o programa é compilado com sucesso, esta informação é indicada na barra de estado.

Na etapa seguinte, o programa compilado deve ser carregado no conversor. Para isto, escolher [Program] / [Compile + download] ou clicar no ícone na barra de ferramentas.

A barra de estado mostra se foi feito o download do programa com sucesso.

O programa IPOS$^{plus®}$ agora é armazenado na memória não-volátil do equipamento.

Também pode ser feito o download dos programas IPOS$^{plus®}$ de um MOVIDRIVE® para outro MOVIDRIVE® utilizando um controle manual DBG. Isto é feito utilizando os parâmetros P807 (Copia MDX -> DBG) e P806 (Copia DBG -> MDX).

18.3 Iniciando / parando programas

Um programa pode ser iniciado depois de ter sido feito o download para o conversor. Para isto, escolher [Run] / [Start] da barra de menu ou clicar no ícone na barra de ferramentas. Uma vez o programa ter sido iniciado, é indicado uma seta verde (indicador do programa) na janela de projeto para destacar a linha do programa a ser processada. O display na barra de ferramentas muda de PSTOP para START.

Para parar os programas no task 1, task 2 e task 3, escolher [Run] / [Stop] da barra de menu ou clicar no ícone na barra de ferramentas. Uma vez o programa ter sido parado, o indicador do programa muda para vermelho e permanece na primeira linha de comando. A indicação de estado para o task 1, task 2, e o task 3 na barra de ferramentas muda de START para PSTOP

Janela de variáveis

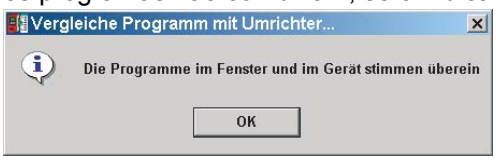
Todas as variáveis e seus conteúdos são indicados na janela de variáveis. Clicar duas vezes em uma variável para mudar o contato diretamente, utilizando o teclado. Pressionar a tecla Enter para adotar o valor novo.

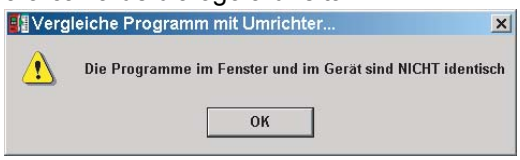
18.4 Comparação entre arquivo e conversor

Utilizar a função comparação do IPOS$^{plus®}$ assembler para comparar um programa assembler carregado na janela Editor com um programa carregado no conversor.

Para acessar a função comparação, escolher [Program] / [Compare with inverter program] da barra de menu ou clicar no ícone na barra de ferramentas.

Se os programas combinam, será indicada a caixa de diálogo abaixo à esquerda. Se os programas não combinam, será indicada a caixa de diálogo à direita.





18.5 Debugger

O debugger integrado é uma ferramenta utilizada para trabalhar através de um programa no modo de passos individuais. Uma vez ter sido feito o download do programa para o conversor, pode-se escolher as três funções.

Run to cursor

Escolher [Run] / [Goto cursor] da barra de menu ou clicar no ícone na barra de ferramentas para funcionar o programa até a posição atual do cursor.

Single step

Escolher [Run] / [Execute single step] da barra de menu ou clicar no ícone na barra de ferramentas para processar a linha do programa a qual o cursor está atualmente posicionado.

Escolher [Run] / [Skip] da barra de menu ou clicar no ícone na barra de ferramentas para pular a linha do programa a qual o cursor está atualmente posicionado. O cursor pula para a próxima linha do programa. Esta função é útil para pular os acessos de função dentro do programa durante testes.

Pode-se parar e reiniciar o programa a qualquer hora durante o debugging clicando no ícone na barra de ferramentas, pressionando a tecla de função F5 ou selecionando [Run] / [Stop] da barra de menu.

Clicar no ícone da barra de ferramentas ou selecionar [Run] / [Start] da barra de menu, para iniciar o programa da posição atual do cursor a qualquer hora, durante o processo de debugging.

Enquanto o programa está funcionando, você pode interrompê-lo pressionando a combinação das teclas Alt+F5. A barra de execução é então movida para o próximo comando a ser processado.

O programa também pode ser interrompido pressionando a tecla F4 [Goto Cursor]. O programa é parado na linha do comando a qual o cursor é posicionado.

18.6 Carregando o programa a partir do conversor

No IPOS$^{plus®}$ assembler você tem a opção de carregar um programa armazenado no conversor. Para isto, escolher [Program] / [Upload] da barra de menu ou clicar no ícone na barra de ferramentas. O processo de carregamento do programa assembler apaga o que você abriu anteriormente.

As linhas de comentário não são salvas no conversor e conseqüentemente são perdidas durante o processo de carregamento do programa.

18.7 Resumo dos ícones

As funções que podem ser acessadas a partir da barra de ferramentas, são indicadas abaixo:

Símbolo	Ítem do menu	Descrição
	File -> Open	Abre um programa
	File -> Save	Salva um programa
	Program -> Compile	Compila um programa
	Program -> Compile + download	Compila um programa e efetua o download no conversor
	Program -> Upload	Carrega um programa do conversor
	Program -> Compare with unit	Compara o programa no editor com o programa no conversor
	Run -> Start	Inicia o programa IPOS$^{plus®}$
	Run -> Stop	Pára o programa IPOS$^{plus®}$
	Run -> Run to cursor	Roda o programa para onde o cursor está posicionado
	Run -> Single step	Roda a etapa simples
	Run -> Skip	Pula uma instrução (comando)
	Edit -> Insert command	Acessa a ferramenta de inserção
	File -> Print	Imprime um programa
	Help -> User manual	Acessa a ajuda online

19 Assembler – Programação

19.1 Básicos

O IPOS$^{plus®}$ assembler é parte do pacote do programa MOVITOOLS®. O programa assembler é inserido sobre várias telas.

Cabeçalho do programa

Para programas do usuário em que são utilizados comandos de posicionamento, inserir as unidades de deslocamento do usuário no cabeçalho do programa.

Task 1 / Task 2 / Task 3

O sistema de controle seqüencial e de posicionamento IPOS$^{plus®}$ permite que o programa do usuário seja dividido em 3 subrotinas (task 1 / task 2 / task 3) que podem funcionar em paralelo e independentes uma da outra.

Comentários

Os comentários podem ser inseridos em qualquer lugar no programa do usuário, como linhas de comando.

Os comentários somente podem ser salvos no PC; eles não são transferidos quando é feito o download do programa para o conversor.

Desvios do programa

Os desvios do programa são possíveis com indicadores de salto (M...) em conjunto com os comandos de salto (JMP... M...). Os indicadores de salto podem ser inseridos antes de qualquer linha de comando.

Sistema de subrotina

As subrotinas podem ser acessadas com o comando CALL (CALL M...). Os indicadores de salto correspondentes (M...) são inseridos antes do primeiro comando da subrotina. A subrotina termina com um comando de retorno (RET). O comando de retorno faz com que o processamento do programa volte para a linha abaixo do comando CALL. As linhas posteriores do programa são então processadas. São possíveis subrotinas aninhadas; mas não deve ser excedido o máximo de 16 camadas.

Observe

As subrotinas nunca devem ser retiradas com um salto no programa principal ou em outra subrotina. Se uma subrotina for retirada condicionalmente, isto deve ser feito saltando para o final (RET) da subrotina.

Rotinas do programa

As rotinas do programa consistem do início da rotina (LOOPB) e do final da rotina (LOOPE). O número de vezes que a rotina é processada é definido no argumento do comando LOOPB. São possíveis rotinas aninhadas; mas não deve ser excedido o máximo de 16 camadas.

Observe

As rotinas do programa nunca devem ser retiradas com um comando de salto. Os comandos de salto são permitidos dentro de uma rotina do programa.

Comandos de posicionamento

O controle de posicionamento IPOS$^{plus®}$ permite realizar o posicionamento ponto-a-ponto com conversores MOVIDRIVE® e MOVIDRIVE® *compact*.

Entradas e saídas digitais / analógicas

As entradas/saídas digitais e analógicas são processadas com variáveis. Além disso, as entradas digitais podem ser avaliadas diretamente utilizando um comando de salto.

Acesso aos valores / parâmetros do sistema

Os parâmetros indicados na seção "IPOS$^{plus®}$ Parameters" como argumentos para os comandos GETSYS e SETSYS, são citados abaixo como valores do sistema. Estes valores do sistema podem ser utilizados conforme a seguir:

- Leitura com o comando **GETSYS**, por ex. corrente ativa e rotação atual.
- Leitura através dos ítens de dados PO.
- Escrita com o comando **SETSYS**, por ex. setpoint fixo.
- Escrita de dados fieldbus através dos ítens de dados PI.
- Os valores do sistema também podem ser lidos e escritos utilizando as variáveis do sistema H458 ... H511 para MOVIDRIVE® A / H458 ... H560 para MOVIDRIVE® B.
- O comando **MOVLNK** permite mudar todos os parâmetros do conversor conectado diretamente ou trocar os parâmetros com outros conversores via RS-485.
- O comando **MOVLNK** permite mudar todos os parâmetros do MQX e MOVIMOT® ou trocar os parâmetros com outros conversores via SBus ou RS-485.

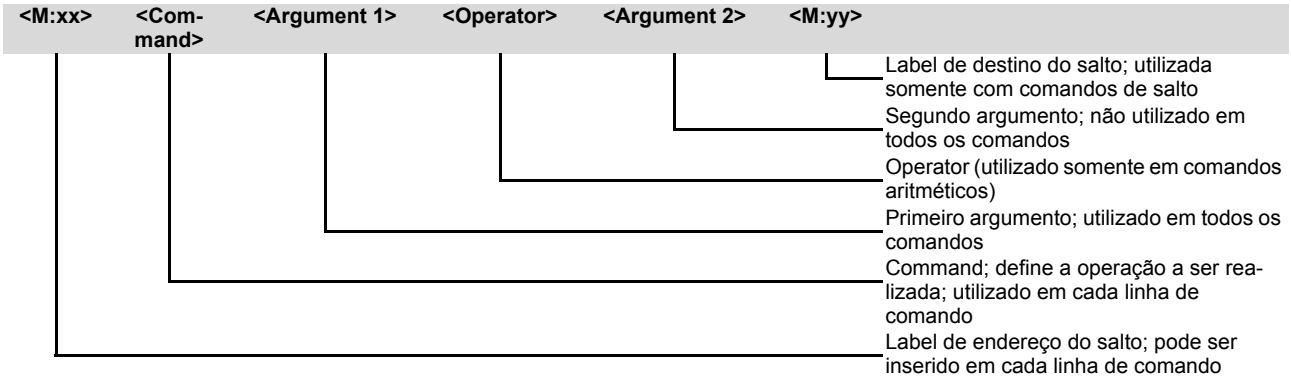
Variáveis

Todas as variáveis (H0 - H1023) podem ser lidas e escritas. As variáveis possuem uma faixa de valor de -2^{31} até $+2^{31} - 1$. Se as variáveis H0 ... H127 são inseridas na lista da variável ou escritas no programa IPOS$^{plus®}$ com o comando "MEM", elas são armazenadas na memória não-volátil assim que elas são inseridas. As variáveis H458 H511 contém os valores da unidade freqüentemente utilizados, que são atualizados ciclicamente (a cada 1 ms). Estas variáveis são citadas abaixo como variáveis do sistema e são explicadas detalhadamente na seção "Resumo das Variáveis do Sistema".

Cuidado ao escrever variáveis do sistema! Os efeitos são descritos na seção "IPOS$^{plus®}$ com Opcionais".

Linha do programa

Comando syntax:



O comando *write* para variáveis e índices distingüi-se entre armazenagem volátil e não-volátil. As variáveis H0 ... H127 podem ser escritas e armazenadas utilizando ambas funções; H128 ... H511 somente podem ser armazenadas na memória volátil.

As variáveis H0 ... H127 são sempre armazenadas na memória não-volátil através do MOVITOOLS® e controle manual. O comando SET de um valor para uma variável em um programa IPOS$^{plus®}$ sempre é armazenado na memória volátil. Para armazenar o estado atual na memória não-volátil, deve ser realizado o comando MEM no programa IPOS$^{plus®}$.

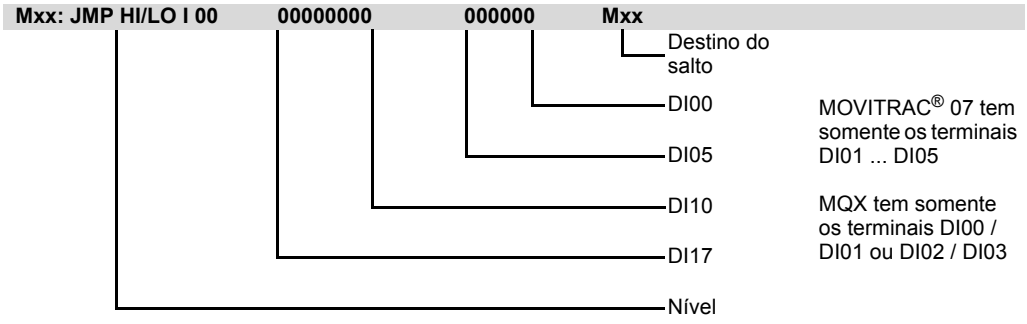
Ao utilizar o comando MEM observe que as variáveis armazenadas na memória não-volátil (H0 127) e todos os parâmetros não são escritos ciclicamente. Isto porque o número de operações com a média de armazenagem EEPROM é restrito a 10^5 operações de armazenagem.

19.2 Entradas/saídas digitais

Entradas digitais

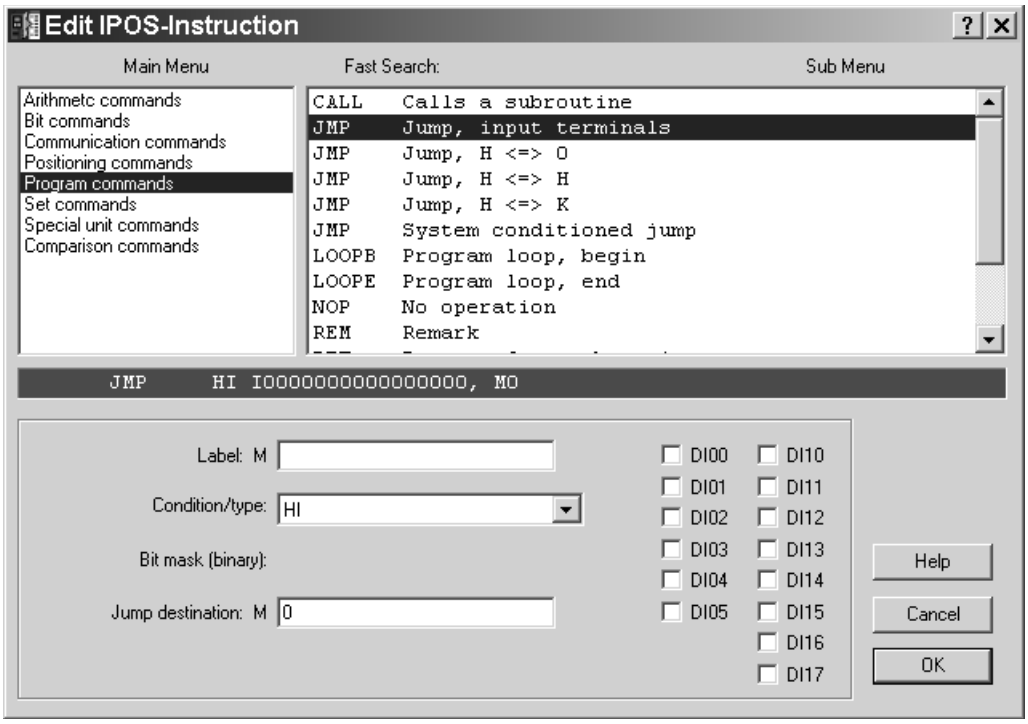
Verificação direta

O nível lógico das entradas digitais pode ser verificado no programa IPOS$^{plus®}$ utilizando comandos de salto. Para isto, na tela de entrada selecione o nível do terminal (HI/LO), que deve conduzir ao comando de salto que está sendo realizado. Os terminais que são utilizados para esta função devem ser identificados com "1" na máscara do terminal. Todos os terminais definidos devem ter o nível do terminal selecionado para cumprir a condição de salto por comando de salto.



Exemplo

Pular para label 20 se as entradas DI03 e DI04 tiverem sinal positivo (1), caso contrário a próxima linha de comando é processada:



10542AEN

Figura 74: Exemplo para um comando de salto dependente do nível do terminal

Interrogação através das variáveis do sistema

O nível lógico das entradas digitais na unidade básica e qualquer opcional instalado são representados ciclicamente nas variáveis do sistema **H483 INPUT LVL** (MOVIDRIVE® A) / **H520 INPUT LVL B** (MOVIDRIVE® B). No processo, cada bit da variável do sistema H483 é atribuído a uma entrada do hardware.

A seleção das variáveis do sistema H483 para MOVIDRIVE® A / H520 para MOVIDRIVE® B, para os terminais da entrada digital é descrita na seção "Variáveis IPOS$^{plus®}$/Resumo das variáveis do sistema."

As entradas digitais no programa IPOS$^{plus®}$ podem ser verificadas utilizando o valor das variáveis H483 / H520. Isto é útil para a verificação das entradas a ser utilizadas para transferir um código binário, por exemplo, para selecionar uma posição da tabela.

Tabela 4: Exemplo de transmissão de um código binário através dos terminais de entrada do MOVIDRIVE® A

Exemplo: Leitura das entradas	**Entradas digitais da unidade**					
Denominações do terminal	DI05	DI04	DI03	DI02	DI01	DI00
Importância	2^5	2^4	2^3	2^2	2^1	2^0
Nível do terminal	1	0	0	0	1	1
Peso	1×2^5	0×2^4	0×2^3	0×2^2	1×2^1	1×2^0
Valor da variável H483[1)]	32	+ 0	+ 0	+ 0	+ 2	+ 1 = 35

1) Se todos os terminais de entrada DIO11A/DIP11A e a palavra de controle 2 estiverem ajustados para nível "0".

Os bornes digitais representados com os bits de valor maior das variáveis H483 / H520 também podem ser verificados utilizando uma combinação dos comandos BMOV e JMP. Este é o caso quando dois opcionais são instalados ao mesmo tempo com a expansão do terminal.

Saídas digitais

Leitura das saídas digitais

Os níveis do terminal das saídas digitais na unidade básica e qualquer opcional instalado são representados ciclicamente nas variáveis do sistema **H482 OUTPUT LVL** (MOVIDRIVE® A) / **H521 OUTPUT LVL B** (MOVIDRIVE® B). No processo, cada bit da variável do sistema H482 é atribuído a uma saída do hardware.

Os níveis individuais do terminal das saídas digitais podem ser avaliados com o comando BMOV no programa IPOS$^{plus®}$. O comando BMOV copia um bit da variável do sistema H482 (OUTPUT LVL) / H521 (OUTPUT LVL B) para qualquer posição do bit (importância) de outra variável. O nível do terminal da saída DO02 é verificado utilizando o exemplo de programa a seguir. Para isto, o bit 1 da variável do sistema H482 é copiado para o bit 0 (importância 2^0) do H200. Isto facilita verificar (0 ou 1) o nível do terminal com um comando JMP.

```
SET      H200 = 0
BMOV H200.0  = H482.1
JMP      H200 == 1 ,Mxx
```

Por outro lado, um ou mais níveis do terminal das saídas digitais pode ser filtrado utilizando uma operação lógica com as variáveis do sistema H 482 (OUTPUT LVL) / H521 (OUTPUT LVL B). O nível do terminal da saída DO02 é verificado utilizando o exemplo de programa a seguir:

```
M1 :  SET      H200 = 2
AND H200 & H482
JMP      H200 == 2 ,M1
```

O resultado da operação AND é escrito para a primeira variável, isto é H200. Portanto, o primeiro argumento deve ser uma variável.

Operação AND de H200 e H482		
H200 = 2	00000000010	(= DO01)
H482 =	11011100110	(= estado atual das saídas digitais)
Result	00000000010	(= o salto é realizado porque H200 = 2)

Ajuste das saídas digitais

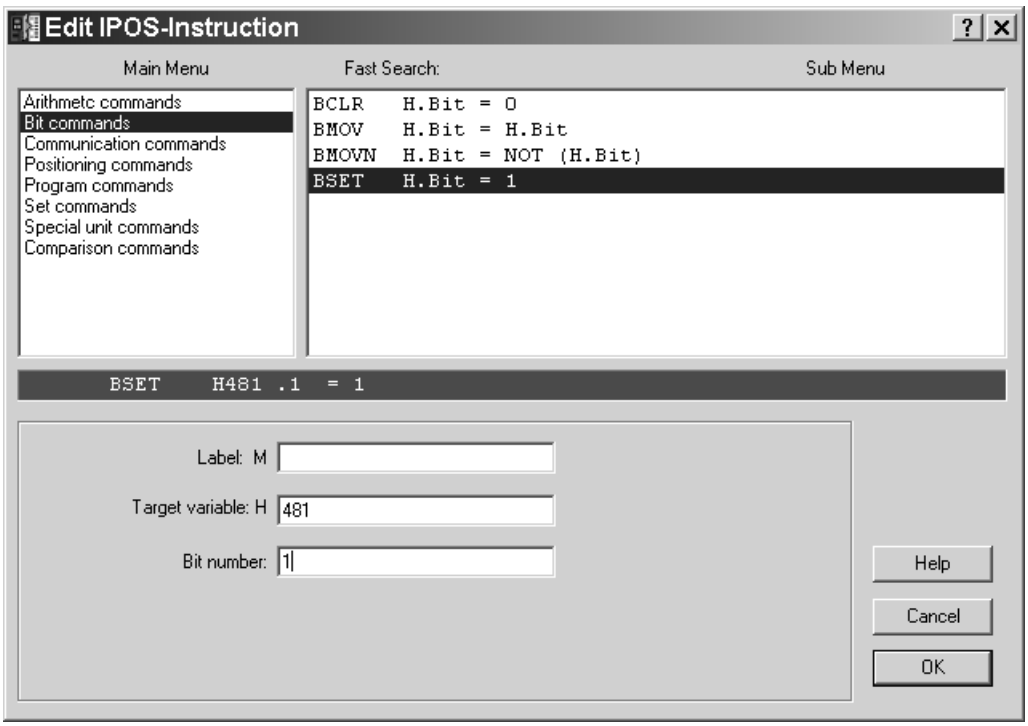
Para ajustar as saídas, os parâmetros da saída digital (parâmetros 620 / 621 (MOVIDRIVE® A), 620 ... 626 (MOVIDRIVE® B), 630 ... 637) devem estar ajustados para "IPOS OUTPUT".

As saídas digitais são ajustadas utilizando as variáveis do sistema:

- **H 480 (OPT. OUT IP)** para opcional DIO11 / DIP11 (DO10 ... DO17)
- **H 481 (STD. OUT IP)** para a unidade básica (DO01 / DO02 (MOVIDRIVE® A) DO01 ... DO05 (MOVIDRIVE® B); DB00 é ajustado para "/Brake". O freio é controlado diretamente pelo firmware. Como resultado, a saída não pode ser escrita.)

Ajuste das saídas individuais

Os **comandos BSET e BCLR** são utilizados para ligar/desligar as saídas individuais. Para isto, o número do bit correspondente ao terminal deve ser inserido como um operando na máscara do comando. No exemplo a seguir, a saída DO01 deve ser ajustada para "1":



10543AEN

Figura 75: Exemplo para ajuste da saída DO01

Resumo dos comandos e parâmetros para ligar / desligar as saídas digitais:

Tabela 5: Comandos para ajuste das saídas digitais

Unidade	Saída	Ajuste (nível 1)	Reset (nível 0)	Parâmetro em "IPOS output"
MOVIDRIVE® A MOVIDRIVE® B	DB00	–	–	Ajustar para "/Brake", isto é, não pode ser programado Controlado pelo firmware.
MOVIDRIVE® A MOVIDRIVE® B	DO01	BSET H481.1 = 1	BCLR H481.1 = 0	P620
	DO02	BSET H481.2 = 1	BCLR H481.2 = 0	P621
MOVIDRIVE® B	DO03	BSET H481.3 = 1	BCLR H483.1 = 0	P622
	DO04	BSET H481.4 = 1	BCLR H481.4 = 0	P623
	DO05	BSET H481.5 = 1	BCLR H481.5 = 0	P624
Opcional	DO10	BSET H480.0 = 1	BCLR H480.0 = 0	P630
	...	...	...	...
	DO17	BSET H480.7 = 1	BCLR H480.7 = 0	P637

Ajuste de várias saídas

É possível ajustar várias saídas digitais ao mesmo tempo, por ex. para saída do número de posição da tabela de codificação binária. Isto é feito escrevendo o valor decimal do número de posição da tabela para as variáveis do sistema H480 ou H481.

Tabela 6: Atribuição da variável do sistema H480 / H481 para bornes de saída digital

Saídas digitais	**Saídas digitais, opcional DIO11A/DIP11A H480**								**Unidade básica das saídas digitais**		
Descrição do borne	DO17	DO16	DO15	DO14	DO13	DO12	DO11	DO10	DO02	DO01	DB00
Bits das variáveis do sistema	7	6	5	4	3	2	1	0	2	1	0
Importância	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^2	2^1	2^0

Exemplo

Número de posição da tabela de saída 11 via DIO11 ("11" necessita 3 saídas; isto é, saídas DO10, DO11 e DO13):

SET H480 = 11

Todas as saídas digitais são reativadas ajustando as variáveis do sistema H480 e H481 para "0".

Isto faz sentido somente se todas as saídas estiverem ajustadas como saídas IPOS$^{plus®}$. As saídas com outros ajustes de parâmetro são escritas pelo firmware e não devem ser modificadas.

SET H480 = 0 Reativar as saídas do opcional DIO11 ou DIP11

SET H481 = 0 Reativar as saídas da unidade básica

19.3 Entradas/saídas analógicas

Tabela 7: Resumo das entradas/saídas analógicas

Entradas/saídas analógicas	**Entradas**						**Saídas**					
	Unidade básica			**Opcional**			**Opcional DIO11**					
Entrada/saída	AI1			AI2			AO1			AO2		
Denominações do terminal	AI11	AI12	AGND	AI21	AI22	AGND	AOV1	AOC1	AGND	AOV2	AOC2	AGND

As entradas analógicas são entradas diferenciais. As entradas/saídas podem ser utilizadas opcionalmente como entradas/saídas de tensão ou corrente.

Tabela 8: Atribuição das faixas de valor para valores da variável

Faixa de valor	**Valor da variável**
– 10 ... 0 ... + 10 V	– 10 000 ... 0 ... + 10 000
0 ... + 10 V	0 ... + 10 000
0 ... + 20 mA	0 ... + 10 000
4 ... + 20 mA	2000 ... + 10 000

A seleção da faixa de valor / valor da variável para saídas analógicas somente é válida se o fator de escala do parâmetro P110 estiver ajustado para 1.

Leitura das entradas/saídas analógicas

O estado das entradas/saídas analógicas da unidade básica e a placa de expansão DIO11 podem ser escritos nas variáveis de sua escolha utilizando o comando **GETSYS**. Inserir primeiro a variável no comando GETSYS, seguida pelo valor do sistema (aqui: **ANALOG INPUTS** ou **ANALOG OUTPUTS**).

A primeira entrada/saída é escrita na variável inserida no comando GETSYS (Hxxx) enquanto a segunda é escrita na variável posterior (Hxxx + 1).

Exemplo

Comando: **GETSYS H310 = ANALOG INPUTS**

H310 contém o valor da entrada analógica AI1

H311 contém o valor da entrada analógica AI2

Escrita das saídas analógicas

O pré-requisito para a escrita é que a saída analógica correspondente (P640 e P643) deve ser ajustada para "IPOS OUTPUT".

As saídas analógicas são ajustadas utilizando as variáveis do sistema **H479 (ANA.OUT IP)** e **H478 (ANA.OUT IP2)**.

Comando: **SET H479 = K** (K = qualquer constante dentro da faixa de valor acima mencionado)

SET H479 descreve saída analógica 1

SET H478 descreve saída analógica 2

20 Assembler – Comandos

Informação geral

- O resultado da operação de cálculo sempre é atribuído ao argumento à esquerda (sempre uma variável). O segundo argumento (variável ou constante) permanece sempre inalterado. O resultado de uma operação matemática sempre é um número inteiro.
- Os exemplos do bit nas variáveis e constantes têm os números 0 ... 31. O bit menos significante tem o número 0.

20.1 Resumo dos comandos

Comandos aritméticos

Este grupo de programa indica todos os comandos aritméticos e lógicos.

Comando	Argumentos	Descrição	Ver
ADD	H + H H + K	Adição aritmética	pág. 273
AND	H & H H & K	Lógica AND	pág. 275
ASHR ARITHMETIC SHIFT RIGHT	H = H (Arithmetic >>) H H = H (Arithmetic >>) K	Deslocamento aritmético à direita	pág. 277
DIV DIVISION	H / H H / K	Divisão	pág. 274
MOD MODULO	H mod H H mod K	Módulo / Divisão restante	pág. 275
MUL MULTIPLY	H * H H * K	Multiplicação	pág. 274
NOT	H = NOT(H)	Negação bit-a-bit	pág. 274
OR	H | H H | K	Lógica OR	pág. 275
SHL SHIFT LEFT	H = H << K H = H << H	Deslocamento bit-a-bit à esquerda	pág. 276
SHR SHIFT RIGHT	H = H >> H H = H >> K	Deslocamento bit-a-bit à direita	pág. 277
SUB SUBTRACT	H – H H – K	Subtração aritmética	pág. 273
XOR EXCLUSIVE OR	H XOR H H XOR K	OR exclusivo	pág. 276

Comandos bit

Comandos para mudança de bits individuais dentro de uma variável. São eles:

- Ajuste/desligamento/movimentação dos bits

Comando	Argumentos	Descrição	Ver
BCLR BIT CLEAR	H.Bit = 0	Desligar o bit	pág. 278
BMOV BIT MOVE	H.Bit = H.Bit	Copiar o bit	pág. 278
BMOVN BIT MOVE NEGATE	H.Bit = NOT (H.Bit)	Copiar o bit e negar.	pág. 279
BSET BIT SET	H.Bit = 1	Ajustar o bit	pág. 278

Comandos de comunicação

Comandos para ampliação de dados de/para outras unidades através das interfaces.

Comando	Descrição	Ver
MOVLNK	Processo acíclico e/ou ampliação dos dados de parâmetro via RS-485 e system bus.	pág. 279
MOVCOM	Transferência de dados de processo cíclica via RS-485 com MQx para MOVIMOT®.	pág. 286
MOVON	Início da transferência de dados de processo cíclica via RS-485.	pág. 288
SCOM SYSTEM BUS COM-MUNICATION	Ampliação de dados de processo cíclica ou acíclica via system bus.	pág. 288
SCOMON SYSTEM BUS COM-MUNICATION ON	Início da ampliação de dados de processo cíclica via system bus.	pág. 294
SCOMST	Início da transferência cíclica para MOVIDRIVE® B.	pág. 295

Comandos de posicionamento

Comandos para posicionamento do conversor:

- Referenciamento
- Posicionamento absoluto/relativo/touch probe

Comando	Descrição	Ver
GO0 GO POSITION 0	Realiza o referenciamento	pág. 296
GOA GO ABSOLUTE	Posicionamento absoluto, variável Posicionamento absoluto, constante Posicionamento absoluto, variável, indireto	pág. 298
GOR GO RELATIVE	Posicionamento relativo, variável Posicionamento relativo, constante Posicionamento relativo, variável, indireto	pág. 299

Comandos do programa

Comandos para controle do programa. São eles:

- Comandos Loop
- Acesso a subrotina
- Controle Task 2
- Comandos de derivação do programa
- Comandos Wait

Comando	Descrição	Ver
CALL	Acessa uma subrotina	pág. 302
END	Fim do programa	pág. 302
JMP JUMP	Jump, borne de entrada Jump, H <=> 0. Jump, H <=> H. Jump, H <=> K. Salto condicionado do sistema	pág. 303
LOOPB LOOP BEGIN	Rotina do programa, início	pág. 306
LOOPE LOOP END	Rotina do programa, final	pág. 306
NOP NO OPERATION	Sem operação	pág. 307
REM REMARK	Comentários	pág. 307
RET RETURN	Final de uma subrotina	pág. 307
TASK2	Ajusta o endereço de partida do task 2	pág. 308
WAIT	Espera por um período especificado	pág. 308

Comandos de ajuste

Comandos para:

- Variáveis de ajuste
- Reações de irregularidade
- Carregando os valores do sistema nas variáveis
- Escrita dos valores do sistema nas variáveis
- Inicializando as rotinas de interrupção

Comando	Descrição / argumentos	Ver
COPY	Cópia bloco-a-bloco das variáveis	pág. 309
GETSYS GET SYSTEM VALUE	H = Valor do sistema	pág. 309
SET	H = H H = K	pág. 312
SETFR SET FAULT REACTION	Ajusta a reação de irregularidade	pág. 312
SETI SET INDIRECT	[H] = H H = [H]	pág. 314
SETINT SET INTERRUPT	Ajusta o endereço de partida da rotina de interrupção	pág. 315
SETSYS SET SYSTEM VALUE	Valor do sistema = H	pág. 317
VARINT	Ajusta o endereço de partida e a estrutura de dados para a interrupção da variável	pág. 319

Comandos especiais do conversor

Comandos para:

- Parar o eixo
- Armazenar variáveis e programas na memória não-volátil no conversor
- Ligar/desligar o touch probe
- Controlar o watchdog

Comando	Descrição	Ver
ASTOP AXIS STOP	Pára o eixo	pág. 321
MEM MEMORIZE	Salva e carrega o programa IPOS$^{plus®}$ e as variáveis.	pág. 322
TOUCHP TOUCH PROBE	Comando Touch probe	pág. 323
WDOFF WATCHDOG OFF	Desliga o watchdog	pág. 326
WDON WATCHDOG ON	Acessa o watchdog em intervalos de tempo	pág. 326

Comandos de comparação

Comandos para comparar variáveis e constantes.

Comando	Argumentos	Ver
ANDL LOGICAL AND	H = H && H	pág. 330
CPEQ COMPARE EQUAL	H = H == H H = H == K	pág. 327
CPGE COMPARE GREATER OR EQUAL	H = H >= K H = H >= H	pág. 327
CPGT COMPARE GREATER THAN	H = H > H H = H > K	pág. 328
CPLE COMPARE LESS OR EQUAL	H = H <= H H = H <= K	pág. 328
CPLT COMPARE LESS THAN	H = H < H H = H < K	pág. 329
CPNE COMPARE NOT EQUAL	H = H != H H = H != K	pág. 329
NOTL LOGICAL NOT	H = NOT(H)	pág. 331
ORL LOGICAL OR	H = H || H	pág. 330

20.2 Comandos aritméticos

Funções aritméticas básicas ADD / SUB / MUL / DIV

As quatro funções aritméticas básicas são realizadas considerando os sinais. Elas também podem ser realizadas com variáveis H e constantes K. O 1º argumento sempre é uma variável H, o 2º argumento pode ser uma segunda variável H ou uma constante K.

ADD

O comando ADD acrescenta uma variável a uma constante e a uma variável (observando os sinais).

Estrutura do comando Mxxx ADD X1 + X2	Mxxx: Label (opcional) X1: Variável (valor e soma) X2: Variável ou constante (valor)
ADD HXX + HYY	A variável HXX é a soma das variáveis HXX e HYY.
ADD HXX + K	A variável HXX é a soma das variáveis HXX e a constante K.

Exemplo 1	`SET H01 = 100`
	`ADD H01 + H01`
	Após o comando ADD, H01 = 200.

Exemplo 2	`SET H01 = 100`
	`ADD H01 + 1`
	Após o comando ADD, H01 = 101.

Exemplo 3	`SET H01 = 2000000000 [0x77359400]`
	`SET H02 = 2000000000 [0x77359400]`
	`ADD H01 + H02 [0xEE6B2800]`
	A faixa de número foi excedida. Após a adição, H01 tem o valor –294967296.

Observe	Se a faixa de número for excedida durante a adição, o resultado é incorreto. Não há mensagem de erro.

SUB / SUBTRACT

O comando SUB subtrai uma variável ou constante de uma variável (observando os sinais).

Estrutura do comando Mxxx SUB X1 – X2	Mxxx: Label (opcional) X1: Variável (minuendo e diferença) X2: Variável ou constante (subtraendo)
SUB HXX – HYY	A variável HXX é o resultado da subtração das variáveis HXX e HYY.
SUB HXX – K	A variável HXX é o resultado da subtração das variáveis HXX e a constante K.

Exemplo 1	`SET H01 = –10`
	`SET H02 = +50`
	`SUB H01 – H02`
	Após o comando SUB, H01 = –60.

Exemplo 2	`SET H01 = +50 [0x00000032]`
	`SET H02 = –2147483600 [0x80000030]`
	`SUB H01 – H02 [0x80000002]`
	A faixa de número foi excedida. Após a subtração, H01 tem o valor –2147483646.

Observe	Se a faixa de número for excedida durante a subtração, o resultado é incorreto. Não há mensagem de erro.

MUL / MULTIPLY

O comando MUL multiplica uma variável com uma constante ou uma variável (observando os sinais).

Estrutura do comando Mxxx MUL X1 * X2	Mxxx: Label (opcional) X1: Variável (fator e produto) X2: Variável ou constante (fator)
MUL HXX * HYY	A variável HXX é o resultado da multiplicação das variáveis HXX e HYY.
MUL HXX * K	A variável HXX é o resultado da multiplicação da variável HXX e a constante K.

Exemplo 1	`SET H01 = –3`
	`MUL H01 * 50`
	Após o comando MUL, H01 = –150.

Exemplo 2	`SET H01 = +50000 [0x0000C350]`
	`SET H02 = +50000 [0x0000C350]`
	`MUL H01 * H02 [0x9502F900]`
	A faixa de número foi excedida. Após a multiplicação, H01 tem o valor –1794967296.

Observe	Se a faixa de número for excedida durante a multiplicação, o resultado é incorreto. Não há mensagem de erro.

DIV / DIVISION

O comando DIV divide uma variável por uma variável ou uma constante (observando os sinais). O resultado é o número predecimal do quociente.

Estrutura do comando Mxxx DIV X1 / X2	Mxxx: Label (opcional) X1: Variável (dividendo e quociente) X2: Variável ou constante (divisor)
DIV HXX / HYY	A variável HXX é o resultado da divisão das variáveis HXX e HYY.
DIV HXX / K	A variável HXX é o resultado da divisão das variáveis HXX e a constante K.

Exemplo	`SET H01 = –13`
	`SET H02 = +3`
	`DIV H01 / H02`
	Após o comando DIV, H01 = –4.

Observe	A divisão por zero conduz a um resultado indefinido. Não há mensagem de erro.

Funções aritméticas auxiliares NOT / MOD

NOT

O comando nega o conteúdo inteiro de uma variável bit-a-bit.

Estrutura do comando Mxxx NOT X1 = NOT (X2)	Mxxx: Label (opcional) X1: Variável (resultado da operação) X2: Variável (valor de saída)
NOT HXX = NOT (HYY)	A variável HXX nega a variável HYY bit-a-bit. Deste modo, a soma hexadecimal da HXX e HYY = 0xFFFFFFFF.

Exemplo	`SET H02 = +1 [0x00000001]`
	`NOT H01 = NOT (H02) [0xFFFFFFFE]`
	Após o comando NOT, H01 = –2.

MOD / MODULO

O comando fornece o restante inteiro quando uma variável for dividida por uma variável ou uma constante. O sinal do resultado é o mesmo que o sinal da primeira variável.

Estrutura do comando Mxxx MOD X1 mod X2	Mxxx: Label (opcional) X1: Variável (dividendo e restante da divisão) X2: Variável ou constante (divisor)
MOD HXX mod HYY	A variável HXX é o restante inteiro após a divisão das variáveis HXX e HYY.
MOD HXX mod K	A variável HXX é o restante inteiro após a divisão da variável HXX e a constante K.

Exemplo 1	`SET H01 = –17`
	`SET H02 = –5`
	`MOD H01 mod H02`
	
	`SET H01 = –17`
	`SET H02 = +5`
	`MOD H01 mod H02`
	
	Após o comando MOD, H01 = –2.

Exemplo 2	`SET H01 = +17`
	`SET H02 = +5`
	`MOD H01 mod H02`
	
	`SET H01 = +17`
	`SET H02 = –5`
	`MOD H01 mod H02`
	
	Após o comando MOD, H01 = +2.

Operações lógicas AND / OR / XOR

AND

O comando AND realiza uma operação AND bit-a-bit entre uma variável e uma segunda variável ou uma constante hexadecimal.

Estrutura do comando Mxxx AND X1 & X2	Mxxx: Label (opcional) X1: Variável (resultado e valor de saída) X2: Variável ou constante (valor de saída)
AND HXX & HYY	A variável HXX é a operação AND bit-a-bit das variáveis HXX e HYY.
AND HXX & K	A variável HXX é a operação AND bit-a-bit da variável HXX e a constante K.

Exemplo 1	`SET H01 = 12` `[0b0000000000001100]`
	`SET H02 = 5` `[0b0000000000000101]`
	`AND H01 & H02` `[0b0000000000000100]`
	
	Após o comando AND, H01 = 4.

Exemplo 2	A posição dentro de uma volta do motor deve ser determinada a partir da posição do encoder do motor.
	`SET H01 = H511`
	`AND H01 & 0xFFF`
	
	Após o comando AND, H01 tem um valor entre 0 e 4095.

OR

O comando OR realiza uma operação OR bit-a-bit entre uma variável e uma segunda variável ou uma constante hexadecimal.

Estrutura do comando Mxxx OR X1 | X2	Mxxx: Label (opcional) X1: Variável (resultado e valor de saída) X2: Variável ou constante (valor de saída)
OR HXX | HYY	A variável HXX é a operação OR bit-a-bit das variáveis HXX e HYY.
OR HXX | K	A variável HXX é a operação OR bit-a-bit da variável HXX e a constante K.

Exemplo	`SET H01 = 12` `[0b0000000000001100]`
	`SET H02 = 1` `[0b0000000000000001]`
	`OR H01 | H02` `[0b0000000000001101]`
	
	Após o comando OR, H01 = 13.

XOR

O comando XOR realiza uma operação XOR bit-a-bit entre uma variável e uma segunda variável ou uma constante hexadecimal.

Estrutura do comando Mxxx XOR X1 XOR X2	Mxxx: Label (opcional) X1: Variável (resultado e valor de saída) X2: Variável ou constante (valor de saída)
XOR HXX XOR HYY	A variável HXX é a operação XOR bit-a-bit das variáveis HXX e HYY.
XOR HXX XOR K	A variável HXX é a operação XOR bit-a-bit da variável HXX e a constante K.

Exemplo	`SET H01 = 65535 [0x00000FFF]`
	`XOR H01 XOR F0F0 hex [0x0000FF0F]`
	Após o comando XOR, H01 = 0xFF0F.

Comandos SHIFT SHL / SHR / ASHR

Os comandos SHIFT são utilizados para movimentar o conteúdo de uma variável bit-a-bit. Todos os bits da variável recebem um significado novo. O número de posições a ser deslocado é especificado no 2º argumento.

SHL / SHIFT LEFT

O comando SHL movimenta o conteúdo de uma variável para a esquerda pelo número de bits especificado em uma variável ou constante. Os zeros são movidos da direita.

Estrutura do comando Mxxx SHL X1 << X2	Mxxx: Label (opcional) X1: Variável (resultado e valor de saída) X2: Variável ou constante (número de operações de deslocamento)
SHL HXX << HYY	Na variável HXX os bits são deslocados HYY posições à esquerda.
SHL HXX << K	Na variável HXX os bits são deslocados por K posições à esquerda.

Exemplo 1	`SET H01 = 31 [0b0000000000011111]`
	`SET H02 = 1`
	`SHL H01 << H02 [0b0000000000111110]`
	Após o comando SHL, H01 = 62.

Exemplo 2	É atribuída uma certa significância binária aos bornes de saída da unidade básica e ao opcional DIO11A. Para utilizar as saídas DO10 ... DO13 para a tabela de posicionamento de maneira útil (4 entradas = 0 ... 15 posições), deslocar a significância das saídas para que o borne com o valor mais baixo DO10 receba uma significância 2^0.
	`SET H01 = 15 [0b0000000000001111]`
	`SET H02 = 6`
	`SHL H01 << H02 [0b0000001111000000]`

SHR / SHIFT RIGHT

O comando SHR movimenta o conteúdo de uma variável para a direita pelo número de bits especificado em uma variável ou constante. Os zeros são movidos da esquerda.

Estrutura do comando Mxxx SHR X1 >> X2	Mxxx: Label (opcional) X1: Variável (resultado e valor de saída) X2: Variável ou constante (número de operações de deslocamento)
SHR HXX >> HYY	Na variável HXX os bits são deslocados HYY posições à direita.
SHR HXX >> K	Na variável HXX os bits são deslocados K posições à direita.

Exemplo 1	`SET H01 = 62 [0b0000000000111110]`
	`SET H02 = 1`
	`SHR H01 >> H02 [0b0000000000011111]`
	Após o comando SHR, H01 = 31.

Exemplo 2	É atribuída uma certa significância binária aos bornes de entrada da unidade básica e ao opcional DIO11A. Para utilizar as entradas DI10 ... DI13 para a tabela de posicionamento de maneira útil (4 entradas = 0 ... 15 posições), deslocar a significância das entradas para que o borne com o valor mais baixo DI10 receba uma significância 2^0.
	`SET H01 = 960 [0b0000001111000000]`
	`SET H02 = 6`
	`SHR H01 >> H02 [0b0000000000001111]`

ASHR / ARITHMETIC SHIFT RIGHT

O comando ASHR desloca o conteúdo de uma variável para a direita pelo número de bits especificados em uma variável ou constante. Os zeros ou uns são deslocados da esquerda, dependendo do sinal do valor original. Isto garante que um sinal negativo seja mantido durante as operações de deslocamento.

Para números positivos, o comando fornece o número predecimal da divisão X1/X2. Para números negativos, o comando fornece o número predecimal da divisão X1/X2 – 1.

Estrutura do comando Mxxx ASHR X1 >> X2	Mxxx: Label (opcional) X1: Variável (resultado e valor de saída) X2: Variável ou constante (número de operações de deslocamento)
ASHR HXX >> HYY	Na variável HXX os bits são deslocados HYY posições à direita.
ASHR HXX >> K	Na variável HXX os bits são deslocados K posições à direita.

Exemplo 1	`SET H01 = 7 [0b0000000000000111]`
	`ASHR H01 >> 2 [0b0000000000000001]`
	Após o comando SHR, H01 = 1.

Exemplo 2	`SET H01 = –7 [0b1111111111111001]`
	`ASHR H01 >> 2 [0b1111111111111110]`
	Após o comando ASHR, H01 = –2.

20.3 Comandos do Bit

Comandos do bit BSET / BCLR / BMOV / BMOVN

BSET / BIT SET

O comando BSET liga um bit dentro de uma variável para 1. As posições do bit na variável têm os números 0 ... 31. O bit menos significante tem o número 0.

Por exemplo, se um bit é ligado na variável do sistema H481 STD.OUT IP, uma saída binária pode ser ligada diretamente. Deve-se ajustar antes, a saída para IPOS OUTPUT nos parâmetros P62x em SHELL.

Estrutura do comando Mxxx BSET HX1.X2 = 1	Mxxx: Label (opcional) X1: Variável final X2: Posição do bit na variável final
BSET HXX.YY = 1	Na variável HXX, o bit YY é ajustado para 1.
Exemplo	SHELL: P621 = IPOS OUTPUT `BSET H481.2 = 1` Após o comando BSET ter sido realizado, o 3º bit é ajustado na variável H481 e saída DO02.
Observe	Se a saída é reservada para uma função diferente (por exemplo, P621 = MOTOR STANDSTILL), o bit é ajustado em H481, mas não a saída binária.

BCLR / BIT CLEAR

O comando BCLEAR ajusta um bit dentro de uma variável para 0. As posições do bit na variável têm os números 0 ... 31. O bit menos significante tem o número 0.

Por exemplo, se um bit é desligado na variável do sistema H481 STD.OUT IP, ele reativa uma saída binária diretamente. Deve-se ajustar antes a saída para IPOS OUTPUT nos parâmetros P62x em SHELL.

Estrutura do comando Mxxx BCLEAR HX1.X2 = 1	Mxxx: Label (opcional) X1: Variável final X2: Posição do bit em uma variável final
BCLEAR HXX.YY = 0	Na variável HXX, o bit YY é ajustado para 0.
Exemplo	SHELL: P621 = IPOS OUTPUT `BCLEAR H481.2 = 0` Após o comando BCLEAR ter sido realizado, o 3º bit é desligado na variável H481 e saída DO02.
Observe	Se a saída é reservada para uma função diferente (por exemplo, P621 = MOTOR STANDSTILL), o bit é desligado em H481, mas não a saída binária.

BMOV / BIT MOVE

O comando BMOV copia um bit de uma variável em um bit em outra variável. As posições do bit de uma variável têm os números 0 ... 31. O bit menos significante tem o número 0.

Estrutura do comando Mxxx BMOV HX1.X2 = HX3.X4	Mxxx: Label (opcional) X1: Variável final X2: Posição do bit em uma variável final X3: Variável fonte X4: Posição do bit em uma variável fonte
BMOV HXX.YY = HZZ.AA	Na variável HXX, o bit YY é ajustado para o valor do bit AA da variável HZZ.
Exemplo 1	`BMOV H2.4 = H7.5` O comando copia o bit 5 da variável H7 no bit 4 da variável H2.

Exemplo 2	`SET H200 = 0`
	`BMOV H200.0 = H473.20`
	`JMP H200 == 1 M01`
	O programa salta para label M01 quando o conversor é referenciado (H473 STAT. WORD).

BMOVN / BIT MOVE NEGATE

O comando BMOVN copia um bit de uma variável para um bit em outra variável, negando-o no processo. As posições do bit de uma variável têm os números 0 ... 31. O bit menos significante tem o número 0.

Estrutura do comando Mxxx BMOVN HX1.X2 = HX3.X4	Mxxx: Label (opcional) X1: Variável final X2: Posição do bit em uma variável final X3: Variável fonte X4: Posição do bit em uma variável fonte
BMOVN HXX.YY = HZZ.AA	Na variável HXX, o bit YY é ajustado para o valor negado do bit AA da variável HZZ.
Exemplo 1	`BMOVN H2.4 = H7.5` O comando copia o bit 5 negado da variável H7 no bit 4 da variável H2.

Exemplo 1	`SET H200 = 0`
	`BMOV H200.0 = H473.20`
	`JMP H200 == 1 M01`
	O programa salta para label M01 quando o referenciamento ainda não tiver sido realizado para o conversor (H473 STAT. WORD).

20.4 Comandos de comunicação

Dados/ampliação dos parâmetros no MOVLNK

MOVLNK

O comando MOVLNK permite que mudanças extensas sejam feitas aos parâmetros do conversor e a todas as outras unidades que podem ser conectadas via system bus ou RS-485. Para garantir a segurança das pessoas e dos sistemas, tomar cuidado especial ao mudar os parâmetros do conversor. Em todos os casos, as precauções de segurança devem poder intervir para neutralizar todos os erros de programação possíveis.

Quando o comando é acessado, o MOVLNK lê e escreve uma vez os dados de processo, as variáveis ou os parâmetros de uma unidade a outra, ou lê ou escreve uma vez as variáveis ou os parâmetros dentro de uma unidade.

Os parâmetros são lidos / escritos utilizando o endereçamento index. Os números index respectivos são indicados no manual "Fieldbus Unit Profile with List of Parameters". Os números index também podem ser indicados no SHELL selecionando o parâmetro e pressionando a combinação das teclas <Ctrl>+<F1>.

Uma interface SBus ou RS-485 pode ser utilizada para comunicação entre 2 unidades.

O MOVILINK pode ser utilizado em um conversor, por exemplo, para salvar a variável de um contador de quantidade protegido contra falhas na alimentação, sem utilizar o comando MEM para salvar nas variáveis não-voláteis. Os dados de processo não podem ser alterados dentro de uma unidade utilizando o comando MOVLINK.

Utilizando o acesso index através do MOVILINK, os valores do conversor definidos pelo usuário que não podem ser acessados com GETSYS / SETSYS também podem ser escritos / lidos do IPOS$^{plus®}$. Este acesso possibilita ao conversor configurar-se no segmento de inicialização.

Antes do comando ser acessado, inicializar as variáveis que o comando utiliza (estrutura do comando). O início desta estrutura do comando é transferido ao comando como um argumento. A estrutura de dados contém o dado a ser escrito ou lido.

Ajustar os parâmetros para comunicação no emissor (mestre) e receptor (escravo). O comando MOVILINK é acessado somente no emissor (mestre).

Os dados de processo somente podem ser trocados entre o MOVIDRIVE® e o MOVIMOT® via RS-485. Neste processo o MOVIDRIVE® sempre é o emissor e o MOVIMOT® sempre é o receptor.

Características	RS-485	SBus
Tempo de funcionamento da rede	30 ms	10 ms (5 ms, somente PD)
Emissor – receptor	Sim	Sim
Multisender[1)]	Não	Sim
Comunicação com MOVIMOT®	sim (somente PD, MOVIMOT® é o receptor)	Não
Observe	Não utilizar a interface Xterminal	Resistores de terminação da rede em ambas extremidades do SBus

1) Diversas unidades conectadas podem iniciar uma comunicação

Estrutura do comando

H+0	**Tipo de rede (interface)** 0 = Reservado 1 = Interface TERMINAL (RS-485#1) USS21A (TERMINAL) **não utilizar!** 2 = Interface S1 (RS-485#2) por ex. para controlar o MOVIMOT® 3 = Reservado 4 = Reservado 5 = SBus, por ex. comunicação eixo-a-eixo para MOVIDRIVE® / MOVIDRIVE® *compact*
H+1	**Endereço individual da unidade final a ser endereçada / Endereço de grupo das unidades finais (receptor) a ser endereçado** Observe em especial, os três endereços a seguir: H+1 = 253 Endereço do conversor H+1 = 254 "Conexão ponto-a-ponto" possível somente com um receptor, independente do seu endereço de ajuste (P810). Os dados do receptor podem ser escritos e lidos. H+1 = 255 "Broadcast", endereçamento simultâneo de todos os receptores conectados, apesar dos seus endereços de ajuste (P810). Os dados do receptor somente podem ser escritos. O **offset de 100** deve ser acrescentado ao endereço de grupo, por ex. 43, se um endereço de grupo **SBus for endereçado com o comando** MOVLNK. Conseqüentemente, o valor a utilizar para a variável H+1 na estrutura do comando é 143.
H+2	**Especificação dos canais de processo (PD) e parâmetro (PARAM) para transferência de dados** 128 = PARAM + 1PD 129 = 1PD 130 = PARAM + 2PD 131 = 2PD 132 = PARAM + 3PD 133 = 3PD 134 = PARAM (sem PD)
H+3	**Serviços de comunicação** 1 = Leitura 2 = Escrita e armazenagem na memória não-volátil 3 = Escrita sem salvar Observe: O usuário tem que assegurar-se que o número máximo de ciclos de escrita permitidos do EEPROM não seja excedido (MOVIDRIVE® A: 100,000 ciclos).
H+4	**O número index do parâmetro ou da variável que deve ser escrita ou lida (ver lista de parâmetros)** (importante se o canal do parâmetro for utilizado.)

H+5	**Número da variável H', a qual o dado de leitura é armazenado ou da qual o dado a ser escrito é obtido**. (A estrutura de dados para H' é descrita detalhadamente abaixo.) Dica: O index de um parâmetro pode ser indicado em Shell colocando o cursor no campo de entrada do parâmetro e pressionando Ctrl+<F1>. O index para variáveis é a variável nr. + 11000 (por exemplo, H13 tem o index 11013).
H+6	**Estado após o comando MOVLNK ter sido realizado. Transferência OK: Zero; número do código de erro se a comunicação não foi bem sucedida.** Ver "Return Codes for Parameter Settings" nos manuais "Serial Communication" ou "Fieldbus Unit Profile with List of Parameters".

Estrutura de dados

H'+0	Contém dados para os serviços de escrita do parâmetro (ver ajustes 2 e 3 para H+3)
H'+1	Contém dados que são lidos para um serviço de parâmetro (ver ajuste 1 para H+3)
	Somente para ampliação dos dados de processo (PD):
H'+2	Dados PO1 da ampliação dos dados de processo
H'+3	Dados PO2 da ampliação dos dados de processo
H'+4	Dados PO3 da ampliação dos dados de processo
H'+5	Dados PI1 da ampliação dos dados de processo
H'+6	Dados PI2 da ampliação dos dados de processo
H'+7	Dados PI3 da ampliação dos dados de processo

Estrutura do comando

Estrutura do comando Mxxx MOVLNK X1	Mxxx: Label (opcional) X1: Iniciando a variável da estrutura do comando
MOVLNK HXX	O comando MOVLNK é realizado utilizando os dados da estrutura do comando iniciando na variável HXX.

Ajustes do parâmetro para o emissor (mestre)

Endereçamento via RS-485: Nenhum ajuste necessário.

Endereçamento via SBus:

Parâmetros	Endereço	Explicação
P816		A taxa de transmissão depende do comprimento do cabo da rede e deve ser o mesmo para o emissor e o receptor.

Ajustes do parâmetro para o receptor

Ampliação dos dados através do canal do parâmetro

Endereçamento via RS-485 (P810 ... P812)

Parâmetros	Endereço	Explicação
P810	0 ... 99	Endereçamento individual (endereço do emissor)
P811	101 ... 199	Endereçamento de grupo (multicast); o emissor pode escrever para todos os receptores ao mesmo tempo, com o mesmo endereço de grupo
P812		A função de monitoração timeout, é útil somente com a transferência de dados cíclica (desativado se ajustado para 0 ms ou 650 ms)

Endereçamento via SBus P88_ e P89_ para MOVIDRIVE® B / P813 ... P816 para MOVIDRIVE® A

Parâmetros	Endereço	Explicação
P881 / P891 P813	0 ... 63	Endereçamento individual (endereço do emissor) (se a opção multisender de SBus for utilizada, isto é, se vários conversores transmitirem o comando MOVLNK ao mesmo tempo, o serviço de comunicação com o endereço final mais baixo (P813) tem prioridade mais alta).
P882 / P892 P814	0 ... 63[1)]	Endereçamento de grupo (multicast); o emissor pode escrever para todos os receptores ao mesmo tempo, com o mesmo endereço de grupo
P883 / P893 P815		Função de monitoração timeout (desativado se ajustado para 0 ms ou 650 ms)
P884 / P894 P816		A taxa de transmissão depende do comprimento do cabo da rede e deve ser o mesmo para o emissor e o receptor.
P886 ... P888 / P896 ... P898 P817... P819		Não pertinente na conexão com o comando MOVLNK

1) Ao utilizar o endereço de grupo, aumentar por 100 o valor de entrada para o endereço final.

Os resistores de terminação SBus devem ser ligados ou conectados na primeira e última estações.

O MOVLNK é um comando *wait*. O próximo comando somente é processado quando o comando MOVLNK tiver sido realizado.

Se dois ou mais comandos MOVLNK são acessados ciclicamente, estes devem ser processados em **um** task. Para MOVIDRIVE® B, isto ocorre preferivelmente no task 2 ou task 3.

Troca de dados através do canal de dados de processo

A comunicação serial deve ser ajustada de acordo com as tabelas acima (endereçamento via RS-485 / SBus) para a ampliação dos dados de processo. São necessários os ajustes adicionais a seguir para utilizar os dados de processo:

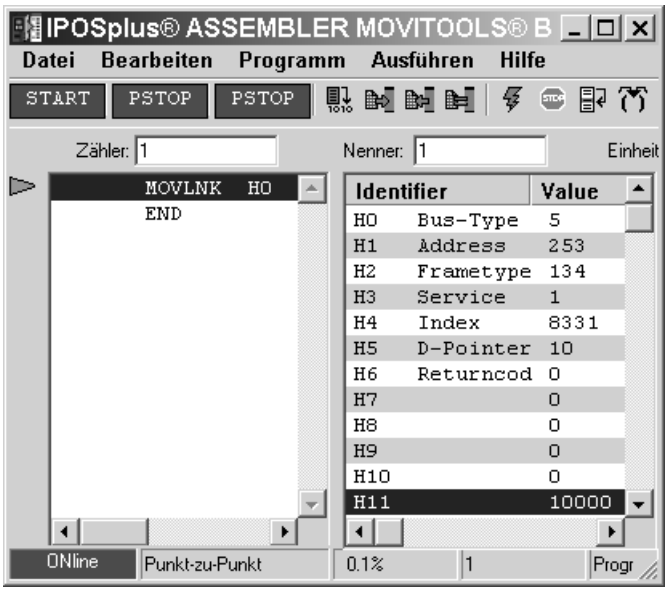
Parâmetros	Explicação
P100	Ajustar a fonte de valor nominal para "RS-485" ou "SBus" (somente se quiser utilizar a especificação do valor nominal através da comunicação de dados de processo)
P101	Ajustar a fonte do sinal de controle para "RS-485" ou "SBus"
P870...876	Descrição dos dados de processo (para descrição mais detalhada, consultar o manual "Fieldbus Unit Profile")

É importante ao utilizar o comando MOVLNK que as variáveis salvas permanentemente (H0...127) e todos os parâmetros não sejam escritos na memória volátil ciclicamente com o serviço de comunicação = 2 porque o número de processos salvos para o meio de memória utilizado (EEPROM) é limitado a 10^5 processos.

Exemplo 1

Leitura de um parâmetro "internal unit" (entrada analógica AI1)

O programa IPOS$^{plus®}$ a seguir e o ajuste de parâmetro são utilizados para ler e indicar o parâmetro P020 com o número index 8331 e então escrevê-lo na variável H011. A estrutura da variável foi inserida aqui na janela de edição para variáveis. A estrutura da variável também pode ser criada no programa utilizando os comandos SET.



H0	Bus type	5	=	SBus (sem significância)
H1	Address	253	=	Próprio endereço
H2	Frame type	134	=	Somente para.
H3	Service	1	=	Leitura
H4	Index	8331	=	Index de P020
H5	D-Pointer	10	=	Valor do indicador de dados é H10

Exemplo 2

Comunicação eixo-a-eixo: Leitura das variáveis de outro conversor via SBus

O valor da variável H005 no eixo receptor é lido e escrito na variável H010 no emissor. Para isto, é necessário ter 2 conversores conectados via SBus e nos resistores de terminação a ser ativados (utilizando a chave DIP S12).

Ajustes acima: Emissor (Mestre) / abaixo: Receptor (escravo)

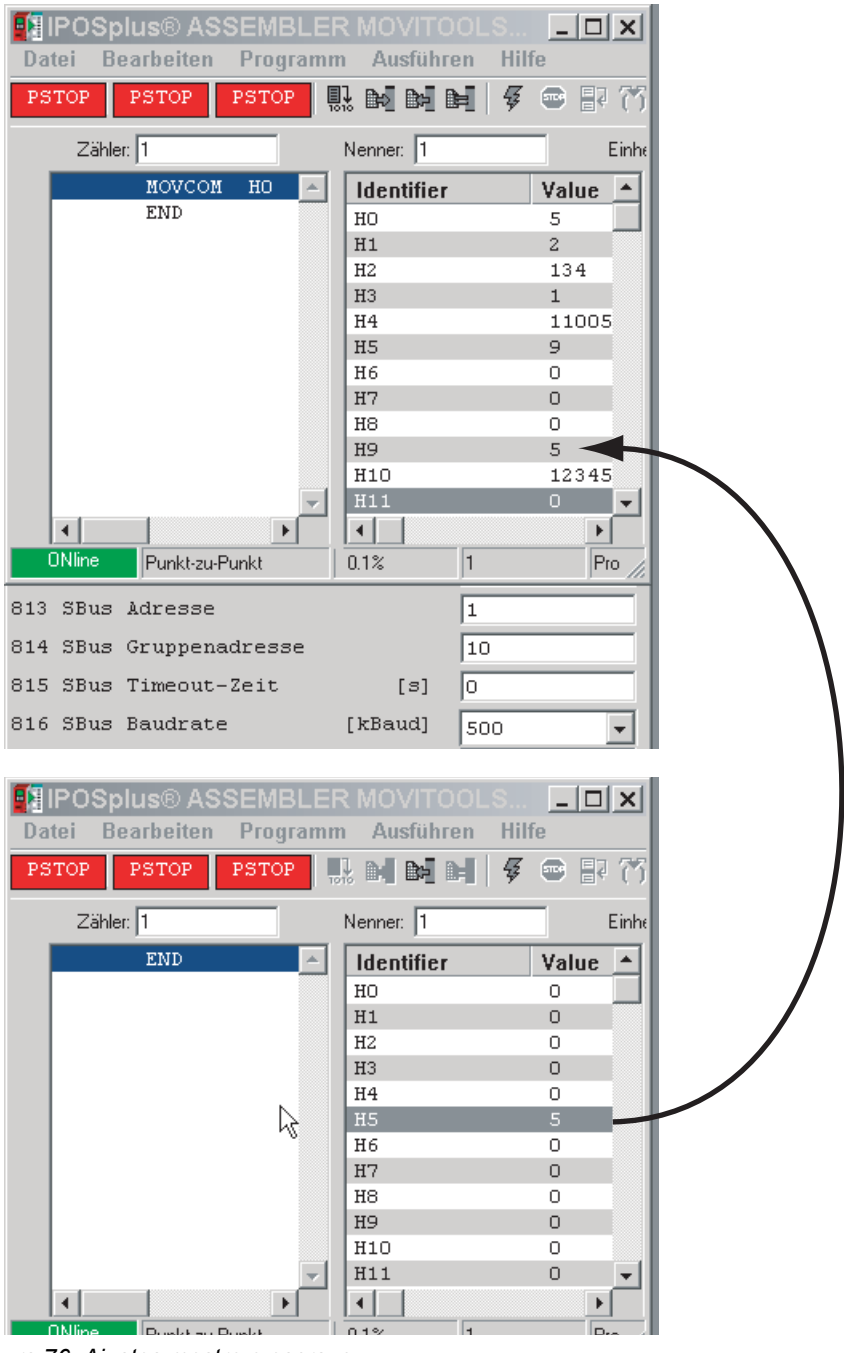


Figura 76: Ajustes mestre e escravo

H0	Bus type	5	=	SBus
H1	Address	2	=	Endereço SBus do receptor (escravo)
H2	Frame type	134	=	Somente parâmetros
H3	Service	1	=	Leitura
H4	Index	11005	=	Index de H5
H5	D-Pointer	9	=	Valor do indicador de dados é H9

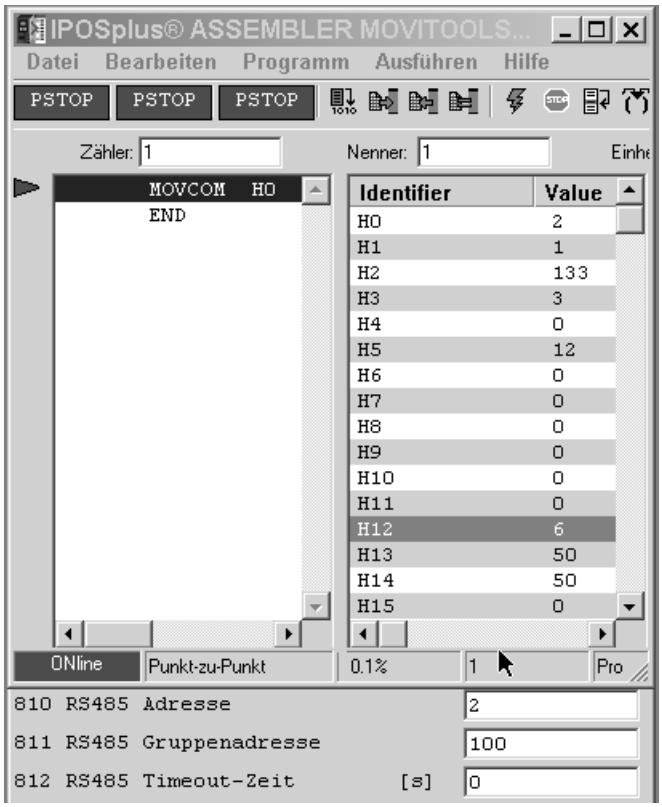
Exemplo 3

Controle de um MOVIMOT® via RS-485 e 3PD

O MOVIMOT® deve ser colocado em funcionamento de acordo com as suas instruções de operação.

A comunicação com MOVIMOT® somente é possível via RS-485. O controle somente é possível via canal de dados de processo com 2PD ou 3PD (palavra de controle e rotação min.).

No exemplo a seguir, o MOVIMOT® é controlado utilizando 3 ítens dos dados de saída de processo (palavra de controle 1, rotação e rampa). Os valores devem ser inseridos nas variáveis H012 ... H014.



H0 Bus type	2	=	RS-485
H1 Address	1	=	Endereço RS-485 do receptor (MOVIMOT®)
H2 Frame type	133	=	3PD
H3 Service	3	=	Escrita sem salvar
H4 Index	0	=	Insignificante para PD
H5 D-Pointer	12	=	Valor do indicador de dados é H12
H12	6	=	PO1 Palavra de controle
H13	50	=	PO2 Rotação em porcentagem
H14	50	=	PO3 Rampa em porcentagem

No MOVIMOT®, a comunicação acíclica desliga a monitoração de timeout. Para a monitoração da comunicação de timeout cíclica na base (independente do tempo de funcionamento do programa), o comando _MovComm é disponível em MQx e nós recomendamos utilizar esta opção.

Uma vez a comunicação cíclica ter sido iniciada com o comando _MovCommOn, é possível somente o comando _MoviLink para endereço 253 (interno). Ao utilizar o comando _MoviLink, o MOVIMOT® não pode mais ser acessado.

MOVCOM

O comando somente pode ser utilizado com MQx, não com MOVIDRIVE®. Os dois comandos MovComm permitem a troca de dados cíclica entre MQx e até 8 conversores MOVIMOT® através da interface RS-485 com o protocolo MOVILINK.

Ampliação de dados via MOVCOM

Os dois comandos MovComm permitem a ampliação de dados cíclica entre MQX e normalmente até 4 conversores MOVIMOT® através da interface RS-485 com o protocolo MOVILINK. _MovCommDef é utilizado para configurar uma conexão de comunicação com MOVIMOT® pelos parâmetros de ajuste, tal como o endereço do conversor. _MovCommOn é utilizado para iniciar a comunicação cíclica. Depois disso, a comunicação cíclica funciona, independente do processamento de comando atual no programa IPOSplus®. A imagem dos dados de processo ampliados é armazenada nas variáveis IPOSplus® e pode ser lida e escrita nestas variáveis. A comunicação cíclica pára quando o programa IPOSplus® é parado.

São permitidas até 8 ligações de comunicação. Observe que o número de ligações de comunicação tem uma grande influência na duração do ciclo da rede de RS-485 e portanto também no tempo de reação do conversor MOVIMOT®. Deve ser considerado aproximadamente 20 ms da duração do ciclo da rede por ligação de comunicação ou estação. O pré-requisito para obter uma duração do ciclo da rede por estação de 20 ms é um cabeamento livre de irregularidade da RS-485.

Se houver um timeout durante a comunicação cíclica, será indicado pela irregularidade 91 "Gateway Sysfault.“ Quando um sinal de retorno é recebido do MOVIMOT®, a mensagem de erro é cancelada automaticamente.

Inserir toda a informação necessária para realizar um comando na estrutura de dados na área da variável com um programa do usuário. O início desta estrutura da variável é o argumento para o comando. A variável é definida pelo **MOVCOM variable name;** no Compilador e tem a seguinte estrutura.

H+0	**Tipo de rede (interface)** ML_BT_S1: S1 (RS-485 #2"
H+1	**Endereço individual ou endereço de grupo para o MOVIMOT® a ser endereçado** 0 ... 99 Endereçamento simples 100 ... 199 Endereçamento de grupo 255 Transmissão
H+2	**Entrada dos dados de processo para transferência de dados** 3 = 2 palavras de dados de processo acíclicas (para MOVIMOT®) 5 = 3 palavras de dados de processo acíclicas (para MOVIMOT®)
H+3	**Número da variável H' a qual os dados de processo são armazenados ou da qual os dados a ser escritos são obtidos**. (A estrutura de dados para H' é descrita detalhadamente abaixo.)
H+4	**Número da variável H’ a qual os dados de parâmetro são armazenados ou da qual os dados a ser escritos são obtidos**. O MOVIMOT® não realiza esta função.

Estrutura da variável dos dados de processo

Estrutura dos dados para H':	
H'+0	Contém o código de erro após a conexão, ou zero se não houver o erro 0x05000002 que indica que a conexão expirou.
H'+1	Dados PO1 da ampliação dos dados de processo
H'+2	Dados PI1 da ampliação dos dados de processo
H'+3	Dados PO2 da ampliação dos dados de processo
H'+4	Dados PI2 da ampliação dos dados de processo
H'+5	Dados PO3 da ampliação dos dados de processo
H'+6	Dados PI3 da ampliação dos dados de processo

Os dados de processo são codificados conforme o MOVILINK.

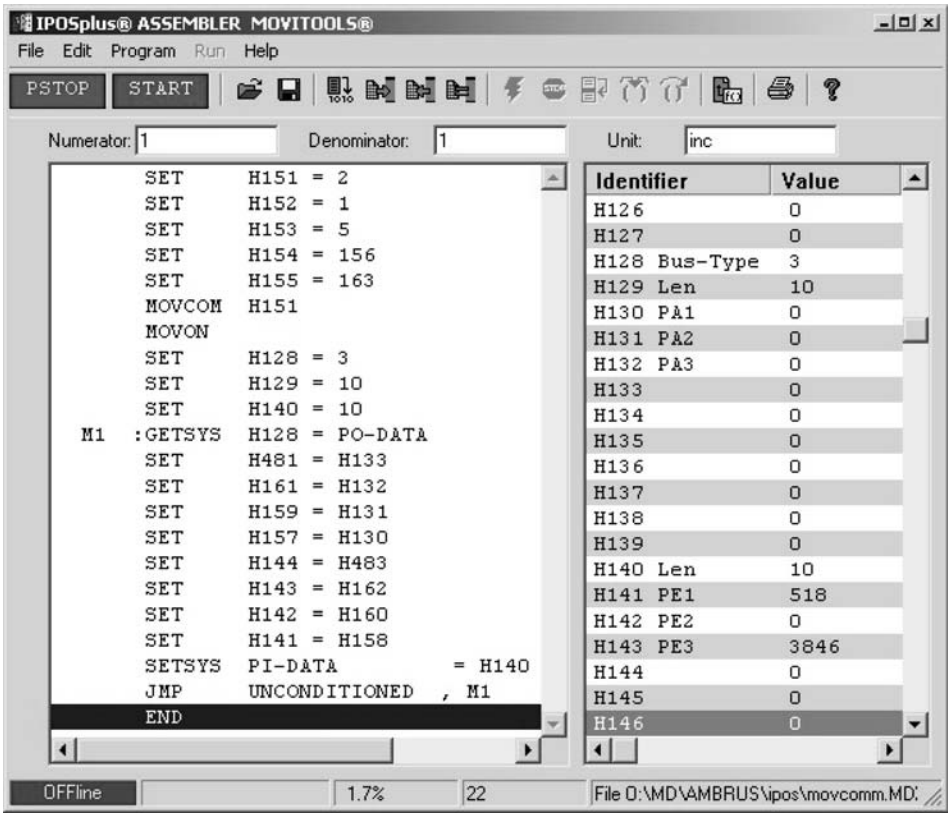
Estrutura da variável dos dados de parâmetro

H+0	Contém o código de erro após o serviço de parâmetro ter sido realizado, ou contém zero se não houver erro. Os erros são codificados conforme o MOVILINK.
H+1	0: Nenhuma ação ou ampliação dos dados de parâmetro está completa. 1: Inicia a ampliação dos dados de parâmetro
H+2	ML_S_RD: Serviço de leitura ML_S_WR: Escrita com armazenagem na memória não-volátil ML_S_WRV: Escrita sem salvar
H+3	Número index do parâmetro a ser revisado ou lido
H+4	Dados de leitura após o serviço de leitura. Dados a ser escritos no caso de um serviço de escrita.

Proceder conforme a seguir, ao fazer os ajustes do parâmetro:

1. Entrada do serviço, index e dados
2. Início do processo de ajuste do parâmetro ajustando StartPar para 1.
3. Espera para que o serviço seja realizado; o final é indicado quando StartPar é ajustado para 0.
4. Avalia ParaResult. Se ocorreu erro, o valor de dados é inválido. Se não ocorreu erro, o serviço foi bem sucedido.

Exemplo de programa



MOVON

O comando não pode ser utilizado com MOVIDRIVE®.

O comando inicia a comunicação cíclica. São ativadas as ligações de comunicação configuradas utilizando o comando MovCommDef. A partir deste ponto, não pode-se mais utilizar os comandos MovCommDef ou MOVILINK. Somente o comando MOVILINK para endereço 253 (interno) pode ainda ser utilizado.

SCOM

Pode ser utilizado um comando SCOM (**S**ystem bus **COM**mmunication) para transferir até 2 variáveis (8 bytes) através do system bus. O comando SCOM inicializa a transferência do objeto e define se o objeto deve ser enviado aciclicamente ou ciclicamente ou se os objetos devem ser recebidos. Nos últimos dois casos, a transferência também deve ser iniciada com SCOMON.

A ampliação de dados somente é possível através do system bus e transfere todo o conteúdo das variáveis. Não é possível a troca de dados dentro do conversor. É utilizado um telegrama CAN padrão (identificador 11 bit) em vez de um protocolo próprio da SEW (MOVILINK), de modo que o sistema possa também se comunicar com produtos não-SEW (ver o manual "MOVIDRIVE® Serial Communication").

De acordo com o princípio consumidor/produtor, cada conversor pode enviar objetos para um ou mais conversores e receber objetos de um ou mais conversores simultaneamente.

O tempo de funcionamento da rede para uma mensagem é de ≤ 2 ms e depende do ajuste da taxa de transmissão. Não é possível a comunicação com MOVIMOT® ou com a interface fieldbus MQ.

Estrutura do comando

Estrutura do comando Mxxx SCOM X1, X2	Mxxx: Label (opcional) X1: TRANSMIT CYCLIC: Envio cíclico RECEIVE: Recebe TRANSMIT ACYCLIC: Envio acíclico X2 Hxx = Inicia a estrutura do objeto para comunicação e dados do usuário

O projeto da estrutura do objeto depende do primeiro argumento X1.

Exemplo

```
SCOM TRANSMIT CYCLIC, H0
```

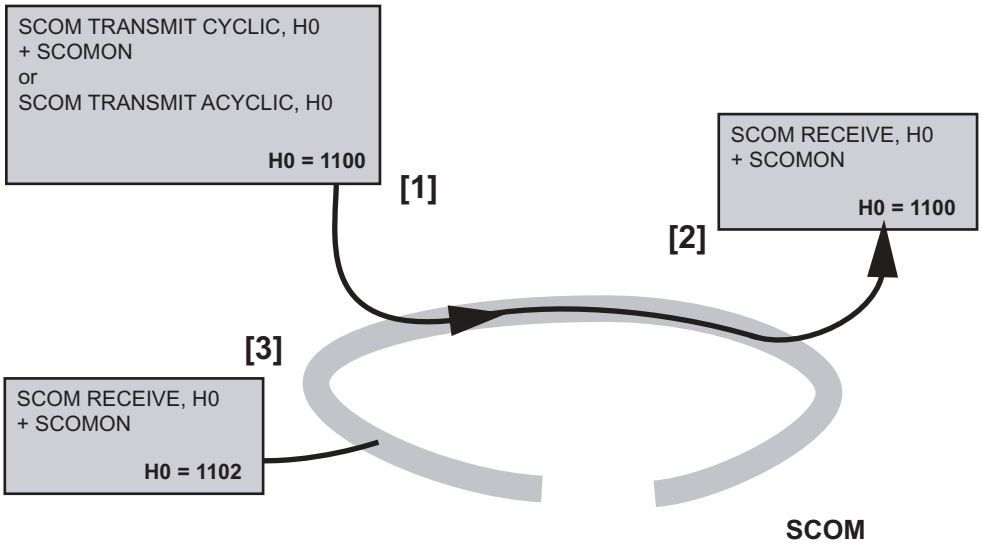
Este comando inicializa a transferência cíclica; a estrutura do objeto inicia em H0.

```
SCOM TRANSMIT ACYCLIC, H10
```

Este comando inicializa uma transferência cíclica one-off; a estrutura do objeto inicia em H10.

```
SCOM RECEIVE, H50
```

O comando inicializa a recepção de dados, a estrutura do objeto inicia em H50.



O conversor [1] envia ciclicamente ou aciclicamente o objeto com o nr. 1100.

O conversor [2] recebe os dados. O conversor [3] ignora os dados do objeto 1100, porém aguarda pelos dados do objeto número 1102.

Ajustes do emissor e receptor

Emissor-	Receptor
Programa IPOS$^{plus®}$ com comando: – SCOM TRANSMIT CYCLIC H SCOMON *and/ or* – SCOM TRANSMIT ACYCLIC H • Ajuste dos parâmetros de comunicação através das variáveis	Programa IPOS$^{plus®}$ com comando: – SCOM RECEIVE H SCOMON • Ajuste dos parâmetros de comunicação através das variáveis • Monitoração de timeout P817
Taxa de transmissão SBus (P816 / P884 / P894) idêntica para emissor e receptor.	
Primeira e última estações físicas: Ajuste do resistor de terminação da rede para S12.	

Observe as regras a seguir ao selecionar o número do objeto:

1. Na rede SBus inteira, um número do objeto pode ser configurado somente uma vez para transmissão.
2. Dentro de um conversor, um número do objeto pode ser configurado somente uma vez para envio ou uma vez para recebimento.

Observe nos ítens 1. e 2. que o firmware reserva alguns números do objeto automaticamente:
- O número do objeto inserido nos parâmetros P885 / P895 (P817 para MOVIDRIVE® A) para sincronização SBus.
- Para comunicação através do protocolo MOVILINK, dependendo do endereço SBus no parâmetro P881 / P891 (P813[1)] para MOVIDRIVE® A) e do endereço de grupo SBus no parâmetro P882 / P892 (P814 para MOVIDRIVE® A), são utilizados os seguintes números do objeto:

– 8 × Endereço SBus + 3	para dados de saída de processo
– 8 × Endereço SBus +4	para dados de entrada de processo
– 8 × Endereço SBus +5	para dados de saída de processo síncronos
– 8 × Endereço SBus +3 + 512	para serviço de pedido do parâmetro
– 8 × Endereço SBus + 4 + 512	para serviço de reação do parâmetro
– 8 × Endereço de grupo SBus +6	para dados do processo de grupo
– 8 × Endereço de grupo SBus + 6 + 512	para pedidos do parâmetro de grupo

- Para comunicação através do protocolo CANopen (em preparação para MOVIDRIVE® B), serão utilizados os números do objeto (identificadores) definidos em DS301 por CANopen.

1) Se o MOVIDRIVE® A for conectado através da placa opcional DFC11A para CANBus, o endereço SBus é feito do ajuste da chave DIP da DFC11A. Neste caso, o endereço de grupo SBus não está ativo.

TRANSMIT CYCLIC:

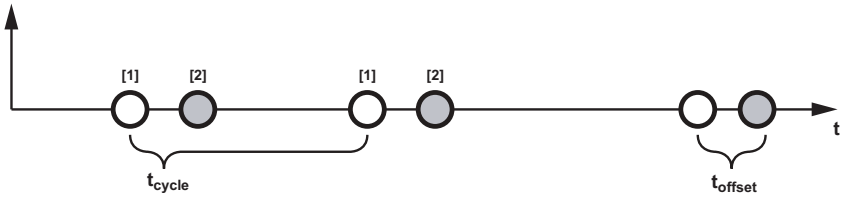
Este argumento inicializa um objeto de dados de quem os dados do usuário são enviados ciclicamente após o comando SCOMON. A variável H do comando SCOM TRANSMIT CYCLIC H define o início da comunicação e os dados do usuário.

A troca de dados cíclica funciona constantemente depois de ter sido iniciada, independente do processamento de comando atual no programa IPOS$^{plus®}$. Se o programa é parado, a transferência de dados pára automaticamente. Uma mudança do objeto de dados somente tornará ativa depois de um reinício do programa IPOS$^{plus®}$ (F5 A/P-STOP / F9 P-Start, ou a rede (operação auxiliar 24 V) desligada e novamente ligada).

O comando Pro SCOM TRANSMIT... pode configurar no máximo um objeto de dados. Os comandos adicionais SCOM TRANSMIT devem ser enviados se os objetos de dados adicionais forem configurados. É necessário somente um comando SCOMON depois de vários comandos SCOM TRANSMIT. Após o primeiro comando SCOMON, não são aceitos outros comandos SCOM TRANSMIT.

O número de objetos que pode ser configurado depende da duração do ciclo (máx. 5 objetos em 1 9 ms, máx. 10 objetos em 10 65530 ms, isto é 15 objetos no total).

Estrutura do objeto

H+0	**Número do objeto (CAN-Bus-ID):** O número do objeto é utilizado para endereçamento do objeto de dados. Um número do objeto somente pode ser atribuído uma vez em um sistema de rede. Os números do objeto do emissor (TRANSMIT) e do receptor (RECEIVE) devem ser os mesmos para a ampliação de dados. Para evitar uma colisão de dados ao utilizar os comandos MOVLNK via SBus, utilizar os números do objeto > 1024 ... 2048.
H+1	**Duração do ciclo** [ms]: Especifica o intervalo de tempo após o qual os dados são novamente enviados. 1, 2 ... 9 10, 20 ... 65530 O valor 0 ms causa uma mensagem de erro a ser emitida no código de retorno.
H+2	**O Offset** [ms] distribui a carga da rede quando vários comandos SCOM TRANSMIT... são utilizados.  Tempos de offset válidos. 0, 1, 2 ... 65534 para duração do ciclo < 10 ms 0, 10, 20 ... 65530 para duração do ciclo ≥10 ms
H+3	**Número de bytes de dados e formato de dados**

Bit	Valor	Função
0...3	0...8	Número de bytes de dados
4...7	0	Reservado
8	0...1	0 = formato MOTOROLA 1 = Formato INTEL **O formato do emissor e do receptor deve ser o mesmo!**
9...31	0	Reservado

H+4	**Número da variável H' a qual deve iniciar os dados a ser enviados.**
H+5	**Resultado (Return Code) do comando SCOM** 0 Capacidade livre da rede em % (valor calculado do conversor) -1 Duração do ciclo incorreta -2 Muitos objetos configurados -3 Sobrecarga da rede
	Certifique-se de que a utilização inteira calculada da rede não exceda 70 % para a troca de dados adicional entre escravos. A utilização da rede é calculada em bits por segundo utilizando a fórmula: Número de mensagens × Bit/mensagem × 1/duração do ciclo Por exemplo, 2 mensagens com 100 bits em ciclo de 1 ms = 200000 bits/s = 200 kBaud Isto resulta na seguinte porcentagem de carga da rede em relação a taxa de transmissão escolhida. Por exemplo, 200 kBaud / 500 kBaud = 40 % < 70 %

TRANSMIT ACYCLIC

Este argumento inicializa um objeto de dados, de quem os dados do usuário são transmitidos imediatamente uma vez. A variável H do comando **SCOM TRANSMIT ACYCLIC H** define o início da comunicação e dados do usuário. Não é necessário um comando SCOMON.

O comando SCOM TRANSMIT ACYCLIC H é utilizado para enviar várias variáveis. Para isto, ajustar o indicador da variável (H+2) de modo devido no programa IPOS$^{plus®}$ antes de acessar cada comando.

Estrutura do objeto

H+0	**Número do objeto (CAN-Bus-ID):** O número do objeto é utilizado para endereçamento do objeto de dados. Um número do objeto somente pode ser atribuído uma vez em um sistema de rede. Os números do objeto do emissor (TRANSMIT) e do receptor (RECEIVE) devem ser os mesmos para a ampliação de dados. Para evitar uma colisão de dados ao utilizar os comandos MOVLNK via SBus, utilizar os números do objeto > 1024 ... 2048.		
H+1	**Número de bytes de dados e formato de dados**		
	Bit	Valor	Função
	0...3	0...8	Número de bytes de dados
	4...7	0	Reservado
	8	0...1	0 = Formato MOTOROLA 1 = Formato INTEL **O formato do emissor e receptor deve ser o mesmo!**
	9...31	0	Reservado
H+2	**Número da variável H' a qual deve iniciar os dados a ser enviados**.		
H+3	Estado do comando de transmissão		
	0	Pronto	
	1	Enviando	
	2	Envio bem sucedido	
	3	Envio de erro	

O programa IPOS$^{plus®}$ espera neste comando até a mensagem ter sido enviada. Se nenhuma outra estação estiver conectada, a mensagem não pode ser enviada. O estado wait somente pode ser terminado pela função de monitoração, por exemplo, de outro task.

RECEIVE

Este argumento inicializa um objeto de dados que contém dados recebidos ciclicamente ou aciclicamente. A variável no argumento do comando SCOM RECEIVE contém o número da variável a qual o dado recebido deve ser armazenado.

O processo de leitura dos dados deve ser iniciado com o comando SCOMON. O processo de leitura dos dados funciona constantemente depois de ter sido iniciado, independente do processamento de comando atual no programa IPOS$^{plus®}$. Após o primeiro comando SCOMON, não são aceitos outros comandos SCOM RECEIVE. Uma mudança do objeto de dados somente tornará ativa depois de um reinício do programa IPOS$^{plus®}$ (F5 A/P-STOP / F9 P-Start ou rede (modo de backup 24 V) desligada e novamente ligada).

Pode-se configurar no máximo 32 objetos de dados para leitura dos dados.

Estrutura do objeto

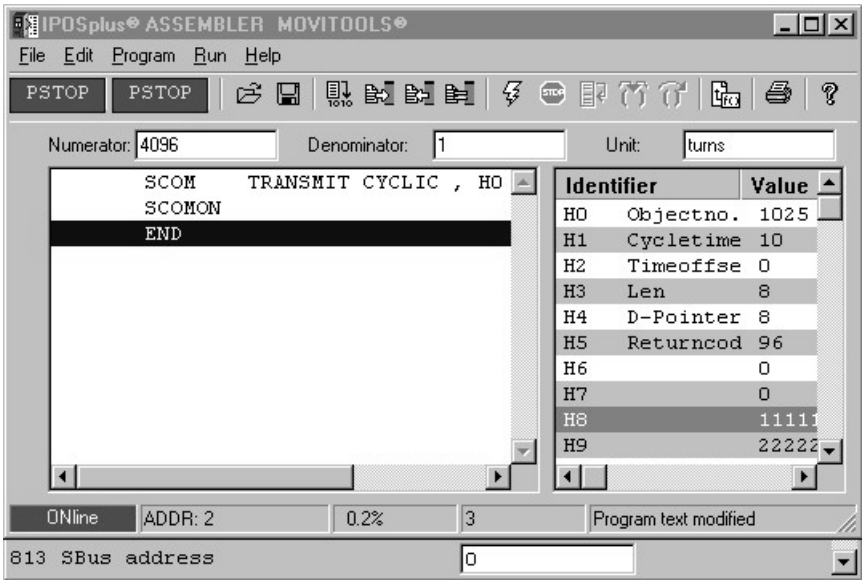
<table>
<tr><td>H+0</td><td colspan="9">Número do objeto: O número do objeto é utilizado para endereçamento do objeto de dados. Os números do objeto do emissor (TRANSMIT) e receptor (RECEIVE) devem ser os mesmos para a ampliação de dados.</td></tr>
<tr><td>H+1</td><td colspan="9">Número dos bytes de dados e formato de dados</td></tr>
<tr><td></td><td>Bit</td><td colspan="3">Valor</td><td colspan="5">Função</td></tr>
<tr><td></td><td>0...3</td><td colspan="3">0...8</td><td colspan="5">Número dos bytes de dados</td></tr>
<tr><td></td><td>4...7</td><td colspan="3">0</td><td colspan="5">Reservado</td></tr>
<tr><td></td><td>8</td><td colspan="3">0...1</td><td colspan="5">0 = Formato MOTOROLA
1 = Formato INTEL
O formato do emissor e receptor deve ser o mesmo!</td></tr>
<tr><td></td><td>9...31</td><td colspan="3">0</td><td colspan="5">Reservado</td></tr>
<tr><td>H+2</td><td colspan="9">Número da variável H' a qual o ponto dos dados recebidos é armazenado
Diferenças nos formatos de dados do usuário MOTOROLA e INTEL:</td></tr>
<tr><td></td><td></td><td colspan="8">Formato MOTOROLA</td><td colspan="8">Formato INTEL</td></tr>
<tr><td></td><td>CAN Data Byte</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td></td><td>Variable</td><td colspan="4">H'+1</td><td colspan="4">H'</td><td colspan="4">H'</td><td colspan="4">H'+1</td></tr>
<tr><td></td><td>Variables byte</td><td>3</td><td>2</td><td>1</td><td>0</td><td>3</td><td>2</td><td>1</td><td>0</td><td>0</td><td>1</td><td>2</td><td>3</td><td>0</td><td>1</td><td>2</td><td>3</td></tr>
</table>

Ao utilizar o comando SCOM, lembre-se que mesmo as variáveis que podem ser armazenadas na memória não-volátil (H0 H127) assim como todos os parâmetros, são escritos somente na memória volátil.

Exemplo 1

Transmissão cíclica de dois valores da variável (H008 e H009) com o comando SCOM do emissor para o receptor às variáveis H005 e H006.

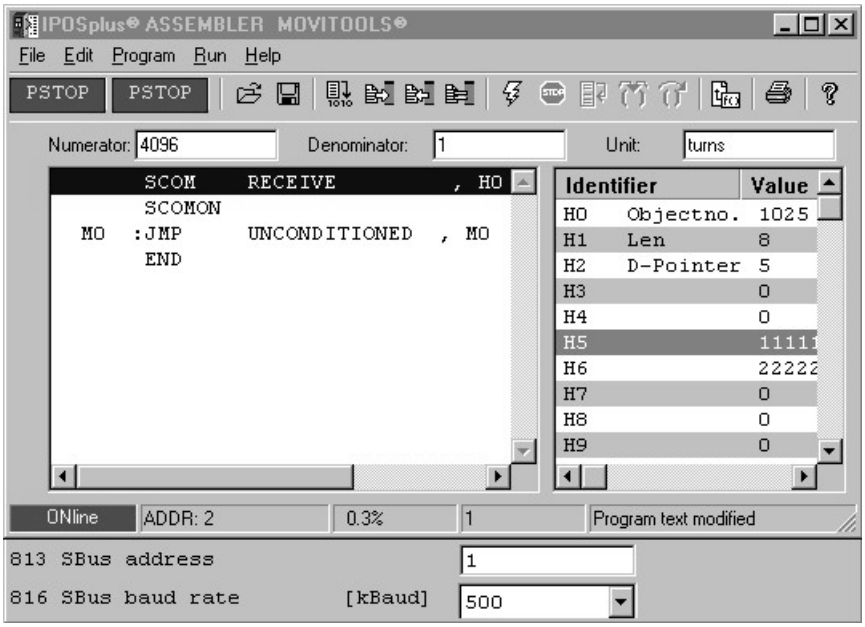
Ajustes do emissor



H0	Object no.	1025	=	Escolha livre
H1	Cycle time	10	=	10 ms
H2	Time offset	0	=	Sem offset
H3	Len	8	=	Comprimento da variável 8 bytes
H4	D-Pointer	8	=	Valor do indicador de dados é H8
H5	Return code	96		
H8		11111	=	Valor enviado

O comprimento de uma variável corresponde a 4 bytes. Isto significa que para um comprimento de dados de 8 bytes, são transmitidas duas variáveis consecutivas.

Ajustes do receptor



H0	Object no.	1025	=	Escolha livre
H1	Len	8	=	Comprimento da variável 8 bytes
H2	D-Pointer	5	=	Valor do indicador de dados é H5
H5		11111	=	Valor enviado

MOVIDRIVE® B / ISYNC ou CAM via SBus

O procedimento de sincronização (sync ID) foi modificado do MOVIDRIVE A. Ao contrário do MOVIDRIVE® A, para MOVIDRIVE® B deve-se assegurar que no programa IPOS$^{plus®}$ do conversor mestre a posição atual seja inicializada **primeiro** e **depois** o objeto sync com SCOM().

SCOMON

System Bus **Com**munication **On**

Este comando ativa a recepção de dados ou a transmissão cíclica dos objetos de dados definidos anteriormente.

O comando SCOM inicializa os objetos de dados com os argumentos RECEIVE (recebe dados) ou TRANSMIT CYCLIC (envia dados ciclicamente).

No MOVIDRIVE® B, o comando foi substituído pelo SCOMST. No entanto, devido a compatibilidade descendente, ainda pode ser utilizado com MOVIDRIVE® B.

Estrutura do comando

Estrutura do comando Mxxx SCOMON	Mxxx: Label (opcional)

SCOMST

Esta instrução inicia ou pára o recebimento dos dados e o envio cíclico dos objetos de dados pré-selecionados via SBus 1 ou SBus 2. A inicialização dos objetos de dados ocorre através da função SCOM com o argumento TRANSMIT CYCLIC ou RECEIVE. O comando é disponível como do MOVIDRIVE® B e pode adotar um dos seguintes valores:

Argumento	Descrição
START ALL	Inicia a comunicação cíclica síncronamente do SBus 1 e SBus 2
STOP ALL	Pára a comunicação cíclica síncronamente do SBus 1 e SBus 2
START1	Inicia a comunicação cíclica do SBus 1
STOP1	Pára a comunicação cíclica do SBus 1
START2	Inicia a comunicação cíclica do SBus 2
STOP2	Pára a comunicação cíclica do SBus 2

Estrutura do comando

Estrutura do comando	
Mxxx SCOMST X1	Mxxx: Label (opcional) X1: Argumento

20.5 Comandos de posicionamento

Referenciamento GO0

GO0

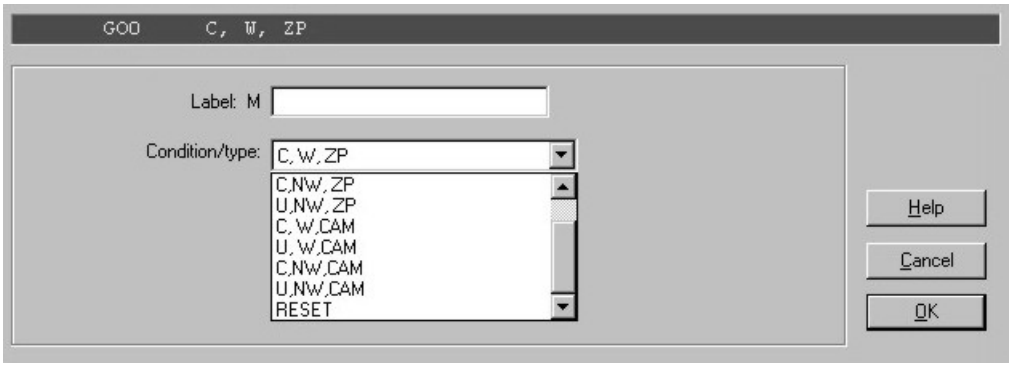
O comando GO0 ativa o referenciamento ou ajusta um encoder absoluto. Fazendo assim, o estado de operação e o display de 7 segmentos muda de "A" (versão aplicação) para "C" (modo de referência). O **Modo de operação P700** não é afetado.

Estrutura do comando

Estrutura do comando Mxxx GO0 X1	Mxxx: Label (opcional) X1: Tipo de referenciamento

O argumento do comando GO0 e os parâmetros P900 ... P903 determinam o comportamento do referenciamento.

O argumento é uma combinação de 3 propriedades características (C/U; W/NW; ZP/CAM) resultando em 8 opções de seleção. O argumento RESET pode ser utilizado para interromper o referenciamento.



C	(conditional)	Realiza o referenciamento somente se o conversor não tiver sido referenciado (isto é, H473, Bit 20 = 0).
U	(unconditional)	Realiza sempre o referenciamento.
W	(wait)	Espera até o eixo ter sido referenciado. Nenhum outro task é realizado neste período.
NW	(non-wait)	O próximo comando é processado durante o referenciamento (recomendado).
ZP	(zero pulse)	Referências para o pulso zero do sinal do encoder (não significante se 903 = 0 ou P903 = 5).
CAM	(reference cam)	Referências para o came de referência (não significante se 903 = 0 ou P903 = 5).
RESET		O referenciamento que iniciou é interrompido (freios da rampa de posicionamento) e o acesso é reativado. Para um eixo de referência, a mensagem "Axis referenced" é reativada e a mensagem "Axis in position" é ajustada.

Ajustes do parâmetro

P60_	Se for utilizado um came de referência, uma entrada deve ser ajustada para a função REFERENCE CAM.
P900	Offset de referência (escreve H498).
P901 / P902	Rotações de referência.
P903	O tipo de referenciamento e o argumento ZP / CAM determinam a condição necessária para finalizar o referenciamento. Exemplo: P903 = 1, GO0 U,W,ZP O pulso zero é avaliado conforme o came de referência.
P904	Não significante para GO0.

O controlador deve ser liberado para ajustar um encoder absoluto utilizando o comando GO0. Por outro lado, o encoder pode ser ajustado sem liberação pelo ajuste do offset P905 para Hiperface® ou os parâmetros P953 ... P955 para um encoder SSI (DIP) .

Para os tipos 3 e 4 e para o ajuste **CAM**, o conversor deve ser referenciado e posicionado à direita próximo a chave fim-de-curso do hardware. Nos sistemas de elevação e em especial o ponto de referência mais baixo, quando o conversor é posicionado no ponto mais baixo, ele pode colidir com a chave fim-de-curso do hardware na ultrapassagem mais fraca. O mesmo perigo aplica-se quando o freio de retenção é aliviado.

Uma maneira de evitar que isto aconteça é posicionar o conversor, uma vez o referenciamento estar completo, bem distante da chave fim-de-curso do hardware (aproximadamente 0.5 ... 1 volta do motor) .

Se as chaves fim-de-curso do software foram ajustadas através do parâmetro P920 / P921, elas somente são monitoradas, uma vez o referenciamento estar completo.

Se o conversor não está conectado a um encoder absoluto ou Hiperface®, o ponto de referência é perdido após ocorrer uma mensagem de erro e o conversor tiver que ser REATIVADO (RESET).

Se um comando de referenciamento de espera é interrompido pela retirada do "/controller inhibit", o código de erro 39 (referenciamento) é indicado.

Os eixos não são colocados em operação, uma vez o sinal ter sido restaurado. O programa IPOS$^{plus®}$ pára neste comando.

Deve ser realizado um reset (entrada digital, fieldbus, SHELL ...). O programa IPOS$^{plus®}$ inicia no começo do primeiro comando.

Posicionamento absoluto GOA/ Posicionamento relativo GOR

O argumento do comando de deslocamento inclui a posição final.

Se a posição final é especificada através de uma variável, o valor somente pode ser inserido em incrementos (baseado em 4096 incrementos/volta do motor). As constantes podem ser inseridas nas unidades de deslocamento do usuário.

Se a função módulo é utilizada para posicionamento, os comandos GOA e GOR não podem ser utilizados; as posições finais são escritas diretamente em H454.

Há dois tipos de posicionamento: absoluto e relativo.

Os eixos giram em um sentido, por exemplo, mesas giratórias, transportador de correias ou alimentadores de rolos geralmente são descritos como módulos de eixo (ver a função módulo P960 ... P963). Neste caso, uma posição mecânica do eixo corresponde a um valor atual H455, independente do número de voltas completadas (pré-requisitos para esta função, ver a descrição da função módulo).

Ajustes do parâmetro para todos os comandos de posicionamento

Parâmetros	Explicação
P913 / P914	Velocidades de deslocamento (podem ser mudadas no programa utilizando SETSYS).
P911 / P912	Rampas de posicionamento (aceleração) (podem ser mudadas no programa utilizando SETSYS).
P915 / P203	Pré-ajustes que podem ser utilizados para influenciar os choques mecânicos.
P933	Suavização de picos (somente com MOVIDRIVE® B).
P916	Tipo de rampa.
P917	Modo de rampa.

GOA / GO ABSOLUTE

Este comando realiza o posicionamento absoluto para a posição especificada no segundo argumento X2. O argumento X2 pode ser uma constante, variável ou uma variável indireta.

A posição final baseada na posição 0 (zero da máquina) é inserida como a distância de deslocamento. A posição final resultante é refletida nas variáveis do sistema H492 (TARGET POSITION).

A mensagem "IPOS in position" é atualizada dentro de um comando GOA ou GOR; isto é, a mensagem pode ser verificada diretamente na próxima linha do programa.

Estrutura do comando

Estrutura do comando Mxxx GOA X1 X2	Mxxx:	Label (opcional)
	X1:	NoWait: O processamento do programa continua enquanto o conversor ainda está em movimento. Isto permite que o programa seja processado ao mesmo tempo como o movimento de deslocamento. Wait: O processamento do programa não continua até a posição atual do conversor ter alcançado a janela de posição P922 da posição final.
	X2:	K = Posição final nas unidades do usuário como uma constante. H = Posição final nas unidades do usuário como uma variável. [H] = Variável, que contém a posição final em incrementos, baseado em 4096 incrementos / volta do motor.

GOR / GO RELATIVE

Este comando realiza o posicionamento relativo para a posição especificada no segundo argumento X2. O argumento X2 pode ser uma constante, variável ou uma variável indireta.

A distância de deslocamento inserida é acrescentada à posição final atual H492 (TARGET POSITION) do conversor e indicada lá.

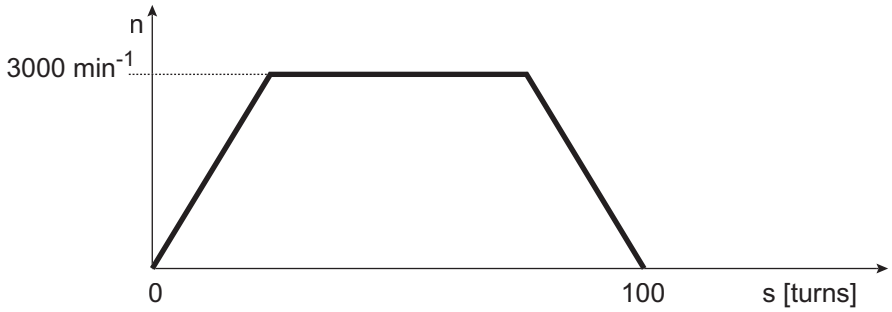
A mensagem "IPOS in position" é atualizada dentro de um comando GOA ou GOR; isto é, a mensagem pode ser verificada diretamente na próxima linha do programa.

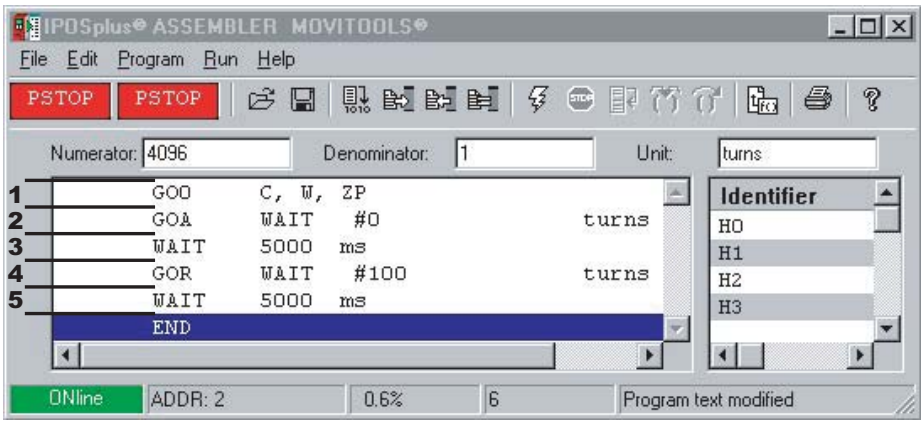
Estrutura do comando

Estrutura do comando Mxxx GOR X1 X2		
	Mxxx:	Label (opcional)
	X1:	NoWait: O processamento do programa continua enquanto o conversor ainda está em movimento. Isto permite que o programa seja processado ao mesmo tempo como o movimento de deslocamento (recomendado). Wait: O processamento do programa não continua até a posição atual do conversor ter alcançado a janela de posição P922 da posição final.
	X2:	K = Posição final nas unidades do usuário como uma constante. H = Posição final nas unidades do usuário como uma variável. [H] = Variável, que contém a posição final em incrementos, baseado em 4096 incrementos / volta do motor.

Exemplo 1

O programa mostrado abaixo faz com que o conversor desloque entre as posições 0 voltas e 100 voltas (entrada no cabeçalho do programa: numerator, denominator, unit). Ocorre um período de espera de 5 segundos quando a posição é alcançada.



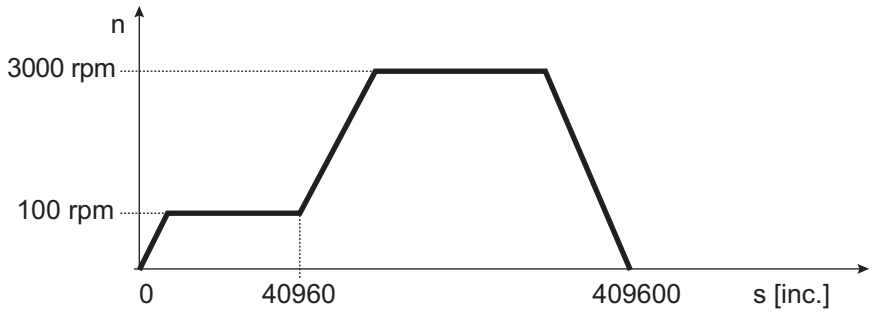


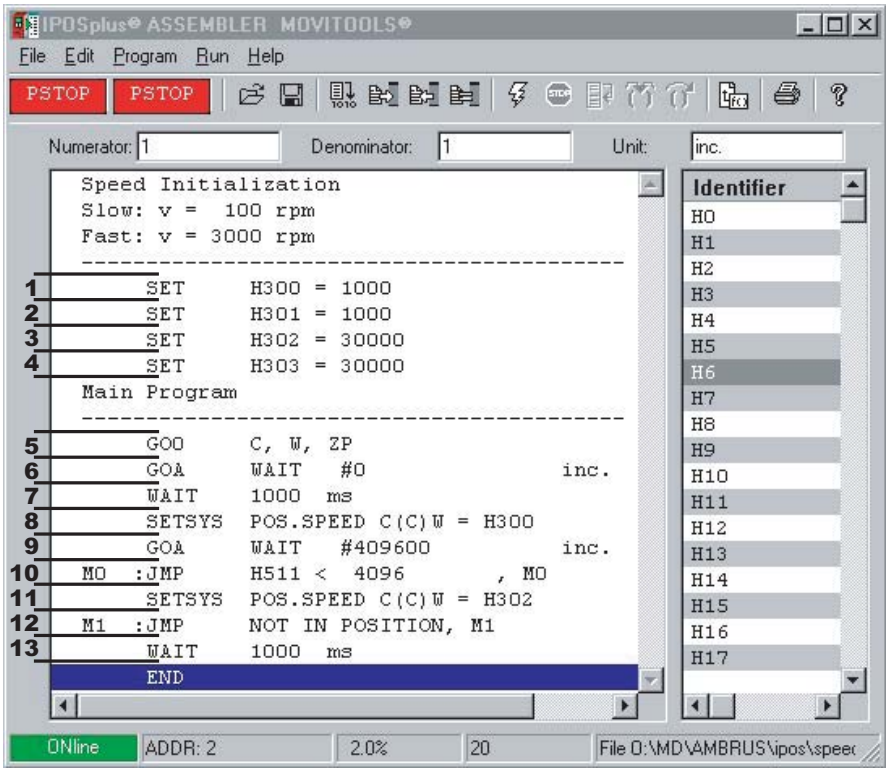
1. Referenciamento
2. Desloca para zero
3. Espera 5 s
4. Desloca para 100
5. Espera 5 s

Uma vez o comando END ter sido realizado, o programa IPOS$^{plus®}$ inicia automaticamente o processamento a partir da primeira linha.

Exemplo 2

O programa mostrado abaixo faz com que o movimento ocorra entre as posições 0 e 409600 incrementos. Ocorre um período de espera de 1 segundo quando a posição é alcançada. A rotação é aumentada de 100 rpm para 3,000 rpm quando o conversor movimenta-se acima da posição 40960. O retorno do deslocamento inteiro ocorre em 3,000 rpm.





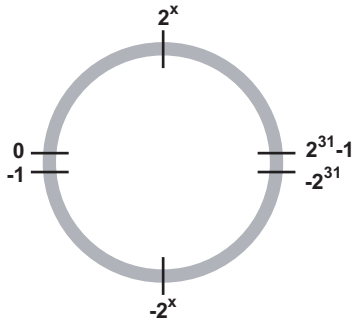
1. Deslocamento lento sentido horário 100 rpm
2. Deslocamento lento sentido antihorário 100 rpm
3. Deslocamento rápido sentido horário 3000 rpm
4. Deslocamento rápido sentido antihorário 3000 rpm
5. Referenciamento
6. Desloca para zero
7. Espera 1 s
8. Ajuste da velocidade lenta
9. Desloca para a posição final
10. Contanto que Actpos Mot (H511) seja menos do que 40960, permanece na linha atual
11. Ajuste da velocidade rápida
12. Permanece na linha atual até o conversor parar
13. Espera 1 s

Posicionamento contínuo

A faixa de deslocamento absoluto do IPOS$^{plus®}$ é limitada aos valores na faixa -2^{31} ... 0 ... 2^{31} –1. Com o comando de deslocamento relativo, a distância de deslocamento máxima de 2^{31} pode ser acrescentada a qualquer posição atual (ver círculo do número).

Um exemplo de posicionamento infinito é mostrado no exemplo do programa no modo JOG.

O comando GOR sempre refere-se a posição final H492. Por exemplo, se o comando GOR 1000 incr. é enviado 100 vezes em um programa, a posição final é ajustada internamente para 100 x 1000 incrementos. O setpoint de posição pode afastar da posição atual do motor se o comando for acessado ciclicamente. O controle IPOS pode então falhar como de um valor crítico $2^{31}/2$ (o conversor gira no sentido oposto).



20.6 Comandos do programa

Comando do programa END

END

O comando END indica o final textual (não lógico) de um programa IPOS$^{plus®}$. O comando END não é um comando IPOS; você não pode apagá-lo.

Acesso a subrotina CALL

CALL

As subrotinas são acessadas com um comando CALL (CALL Mxx). Os indicadores de salto correspondentes (Mxx) são inseridos na frente do primeiro comando na subrotina. A subrotina termina com o comando RETURN (RET). O comando RETURN faz com que o processamento do programa salte de volta para a linha abaixo do comando CALL. As linhas seguintes do programa serão então processadas. Também é possível ter acesso a subrotinas aninhadas (profundidade máxima do aninhamento: 32 níveis).

Estrutura do comando

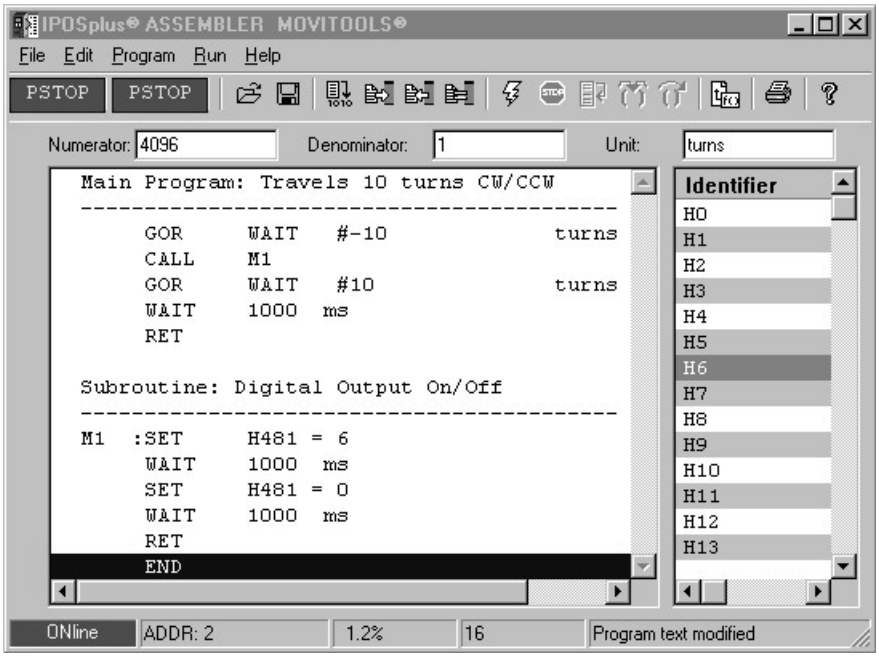
Estrutura do comando Mxxx CALL Myyy	Mxxx: Label (opcional) Myyy = Label a qual a subrotina inicia.

Observe

Não saia das subrotinas saltando para um programa principal ou para outra subrotina. A saída condicional da subrotina deve ser realizada saltando para o final da subrotina.

Exemplo

O programa principal posiciona o conversor 10 voltas sentido antihorário, depois do que há um acesso a subrotina (CALL M1). Ajustar 2 saídas da unidade básica para 1 s (os parâmetros de saída devem ser ajustados para "IPOS-OUTPUT"). O salto de volta para o programa principal (RET) ocorre em seguida e o comando de posicionamento GOR WAIT #10 é processado.



Comandos de salto JMP

JMP / bornes

O programa salta para a label especificada se todos os bornes de entrada marcados na máscara forem ajustados para nível 1 ou nível 0 (relação AND).

Os bits 0 ... 5 indicam os bornes da unidade básica, bits 6 ... 13 os bornes da placa opcional (DIO11A). A máscara é criada inserindo os níveis do borne diretamente na janela de entrada.

Estrutura do comando

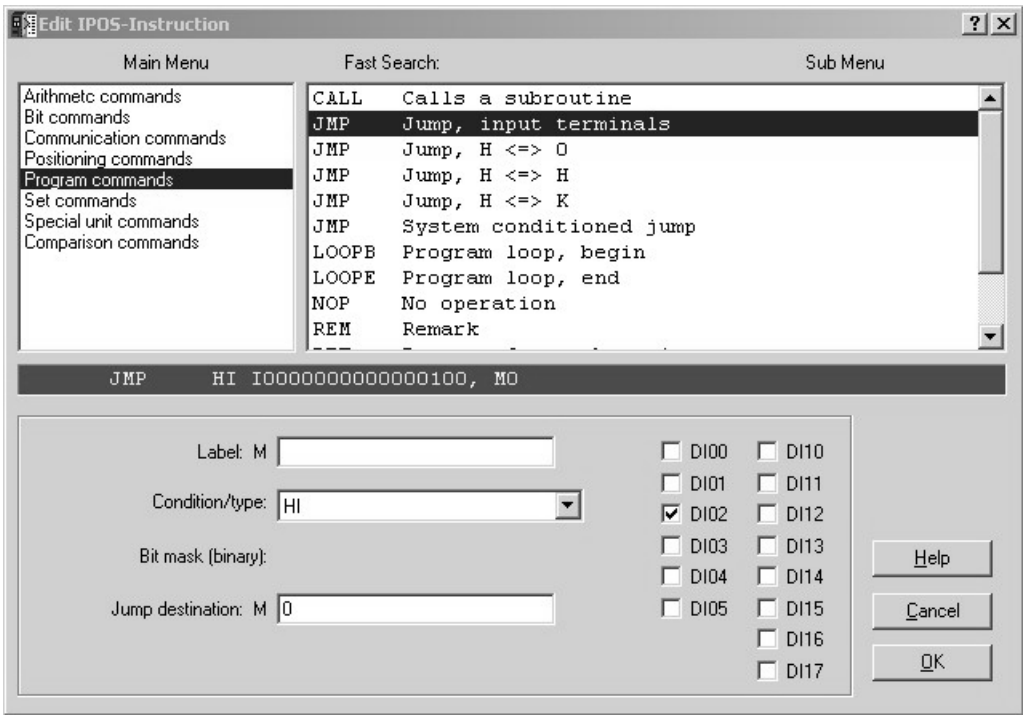
Estrutura do comando Mxxx JMP X1 X2, Myyy		
	Mxxx:	Label (opcional)
	X1:	HI= Salta se os bornes de entrada marcados na máscara forem ajustados para nível 1. LO= Salta se os bornes de entrada marcados na máscara forem ajustados para nível 0.
	X2:	Ixxx ... = Máscara para os bornes de entrada.
	Myyy:	Salta a label a qual o programa deriva.

Exemplo

```
JMP HI I 0000000000000011, M03
```

Após o comando JMP ter sido realizado, o processamento continua da label M03 se os bornes de entrada DI00 e DI01 estiverem ajustados para nível 1.

.

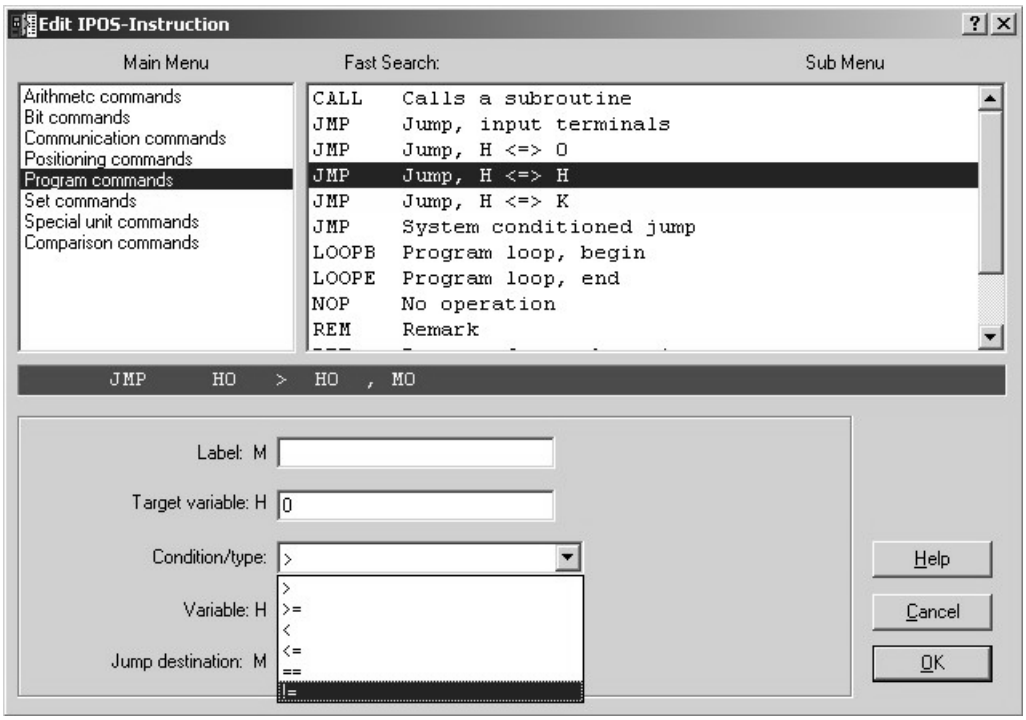


JMP / comparação

O comando JMP faz com que o programa salte para uma label especificada quando a comparação resulta em um comando verdadeiro.

Estrutura do comando

Estrutura do comando Mxxx JMP X1 OP X2, Myyy	Mxxx:	Label (opcional)
	X1:	Variável
	OP:	Operador: > / >= / < / <= / = / !=
	X2:	H = variável K = constante 0 = zero (na comparação com zero, somente os operadores = and != são possíveis).
	Myyy:	Label de salto a qual o programa salta se a condição estiver realizada.

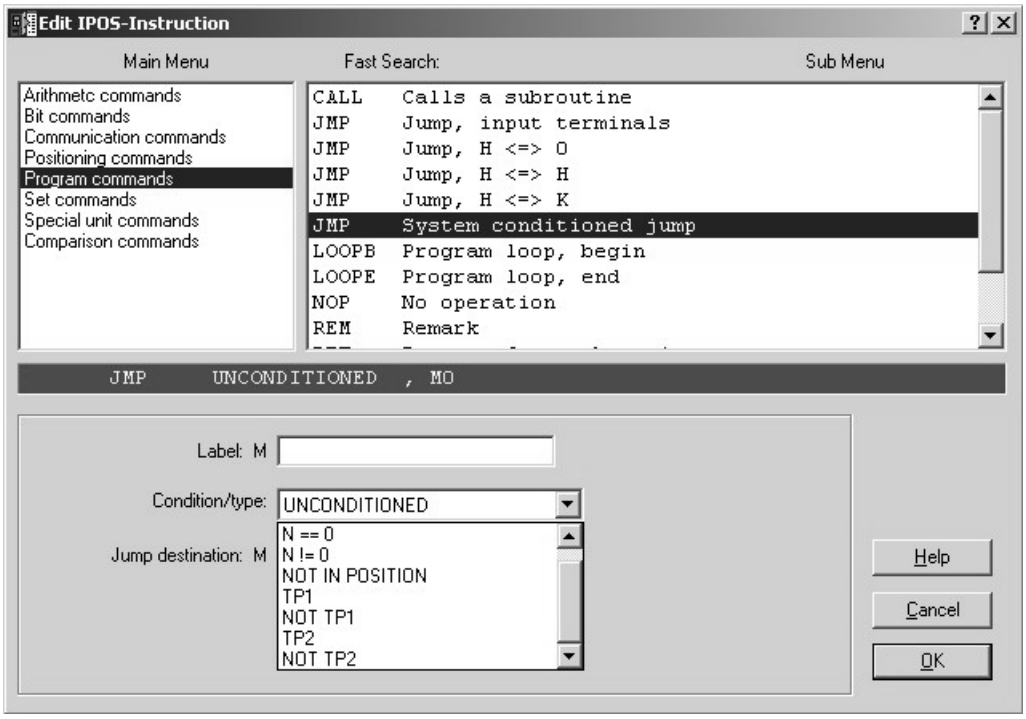


JMP / sistema

Salta para a label indicada no comando se a operação de comparação for realizada.

Estrutura do comando

Estrutura do comando Mxxx JMP X1, Myyy		
	Mxxx:	Label (opcional)
	X1:	UNCONDITIONED: Salto incondicional. N == 0: Salta se a rotação é igual a zero. N != 0: Salta se a rotação é diferente de zero. NOT IN POSITION: Salta se sem posição TP1: Salta se houver uma mudança de pulso no terminal touch probe DI02. NOT TP1: Salta se não houver uma mudança de pulso no terminal touch probe DI02. TP2: Salta se houver uma mudança de pulso no terminal touch probe DI03. NOT TP2: Salta se não houver uma mudança de pulso no terminal touch probe DI03.
	Myyy:	Label de salto a qual o programa salta se a condição for realizada.



Os valores do sistema indicados podem ser verificados diretamente por meio de um comando JMP. Outros valores do sistema são disponíveis como variáveis do sistema ou devem ser lidos utilizando o comando GETSYS e processados posteriormente.

Comandos de rotina LOOP

LOOPB / LOOP BEGIN

Este comando em combinação com um comando LOOP cria uma rotina do programa. O número de ciclos de rotina (> 0) é especificado como uma constante. A rotina termina no comando LOOP associado. As rotinas do programa podem ser aninhadas.

Se o número de ciclos de rotina deve ser variável, deve ser utilizado um comando JMP em vez de um comando LOOP e a condição da variável deve ser verificada em cada ciclo.

Estrutura do comando

Estrutura do comando Mxxx LOOPB X1	Mxxx:	Label (opcional)
	X1:	Número de ciclos de rotina (máximo 256)

LOOP / LOOP END

Este comando especifica o final de uma rotina do programa que foi iniciada utilizando o comando LOOPB.

Estrutura do comando

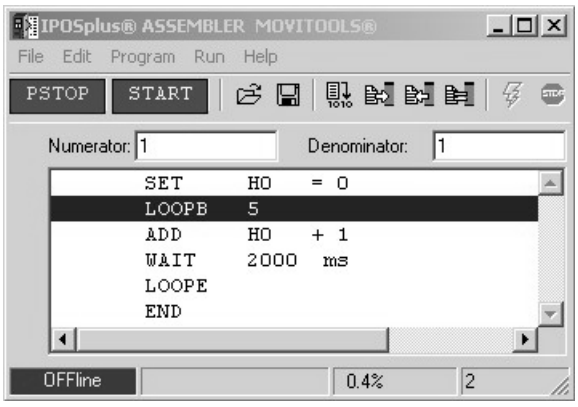
Estrutura do comando Mxxx LOOPE	Mxxx:	Label (opcional)

Observe

Não sair das rotinas do programa com um comando de salto. Os comandos de salto e as subrotinas são permitidos dentro de uma rotina do programa.

Exemplo

No exemplo abaixo, a variável H0 é incrementada de 0 até o valor 5 em 5 ciclos de rotina. O processamento do programa inicia novamente com o comando SET H0 = 0 após 5 ciclos de rotina.



Nenhuma operação NOP / remark REM / return RET / TASK / TASK2 / wait WAIT

NOP / NO OPERATION

Nenhuma operação é realizada. Este comando pode ser utilizado, por exemplo, para obter tempos de espera com base na duração de ciclo do comando. No MOVIDRIVE® A, por exemplo, quer dizer 1 comando/ms no task 1.

Estrutura do comando

Estrutura do comando Mxxx NOP	Mxxx:	Label (opcional)

REM / REMARK

O comando REM acrescenta uma linha de comentário ao programa. As linhas de comentário não podem ser salvas no conversor. Todas as linhas de comentário são perdidas após o DOWNLOAD do programa seguido por um UPLOAD.

As linhas de comentário somente podem ser salvas nos arquivos do programa no PC.

Estrutura do comando

Estrutura do comando REM X1	X1:	Qualquer caracter

RET / RETURN

O comando RET termina uma subrotina (ver o comando CALL) e salta de volta para o programa o qual a subrotina foi acessada. Em um programa principal, o comando RET faz com que um salto volte ao início do programa principal.

Estrutura do comando

Estrutura do comando Mxxx RET	Mxxx:	Label (opcional)

TASK

Este comando é utilizado para definir o endereço de partida do task 2 e task 3 e para iniciá-los ou pará-los com o argumento X1 (START/ STOP); isto é, a palavra de controle do task é escrita. A palavra de controle e o endereço de partida são ajustados para 0 quando a energia é ligada, isto é o Task2 é desativado.

Este comando somente é disponível como do MOVIDRIVE® B.

Estrutura do comando

Estrutura do comando Mxxx TASK X1 Myyy	Mxxx:	Label (opcional)
	X1:	TASK2 STOP: Stop task 2. TASK2 START: Start task 2. TASK3 STOP: Stop task 3. TASK3 START: Start task 3.
	Myyy:	Label a qual o task inicia.

Exemplo

```
TASK TASK2 START M03
```

O task 2 é iniciado neste comando e o primeiro comando após a label M03 é processado paralelo ao task 1.

TASK2

Este comando é utilizado para definir o endereço de partida do TASK2 e para iniciá-lo ou pará-lo com o argumento X1 (START/ STOP); isto é, a palavra de controle do TASK2 é escrita. A palavra de controle e o endereço de partida são ajustados para 0 quando a energia é ligada, isto é o TASK2 é desativado.

No MOVIDRIVE® B, o comando foi substituído pelo TASK. No entanto, devido a compatibilidade decrescente, ele ainda pode ser utilizado com MOVIDRIVE® B.

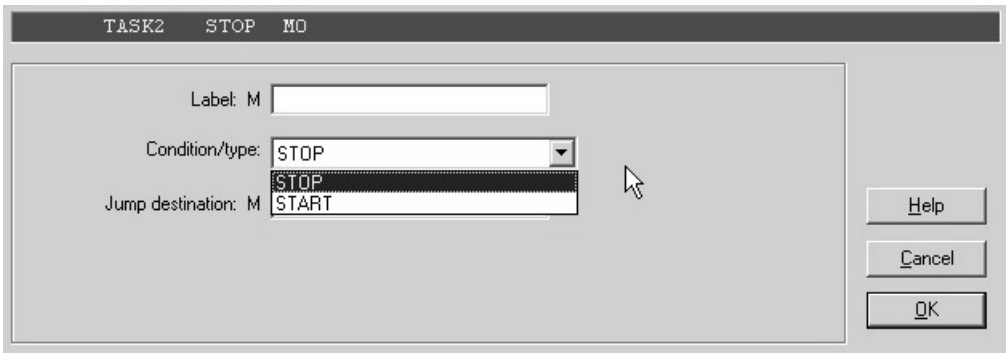
Estrutura do comando

Estrutura do comando Mxxx TASK2 X1 Myyy		
	Mxxx:	Label (opcional)
	X1:	STOP: Stop task 2. START: Start task 2.
	Myyy:	Label a qual o task 2 inicia.

Exemplo

```
TASK2 START M03
```

O task 2 é iniciado neste comando e o primeiro comando após a label M03 é processado paralelo ao task 1.



WAIT

O comando WAIT espera pelo comprimento do tempo especificado em ms no argumento e então continua o processamento do programa para esta tarefa uma vez que o tempo acabou.

Estrutura do comando

Estrutura do comando Mxxx WAIT X1		
	Mxxx:	Label (opcional)
	X1:	Tempo de espera em ms, 0 ... 32767.

Se o tempo de espera deve ser variável, deve-se inicializar um timer (H487 ... H489) em vez de um comando WAIT e programar uma rotina com o comando JMP até o timer ter expirado.

Exemplo

```
SET H0 = 20000
SET H489 = H0
M01: JMP H489 != 0, M01
```

20.7 Comandos de ajuste

Variáveis de cópia COPY

COPY

O comando COPY copia o número de variáveis sucessivas especificadas no 3º argumento. O segundo argumento do comando COPY indica o número da primeira variável fonte; o primeiro argumento indica o número da primeira variável final. Podem ser copiadas até 10 variáveis utilizando um comando COPY.

Estrutura do comando

Estrutura do comando Mxxx COPY X1 = X2, X3	Mxxx:	Label (opcional)
	X1:	Hxxx = Número da primeira variável final.
	X2:	Hyyy = Número da primeira variável fonte.
	X3:	K = Constante (número das variáveis a ser copiadas, 1 ... 10).
	Myyy:	Label de salto a qual o programa salta se a condição for realizada.

Exemplo

O comando `COPY H2 = H20, 3` corresponde a seqüência de comando:

```
SET H2 = H20
SET H3 = H21
SET H4 = H22
```

Leitura dos valores do sistema GETSYS

GETSYS / GET SYSTEM VALUE

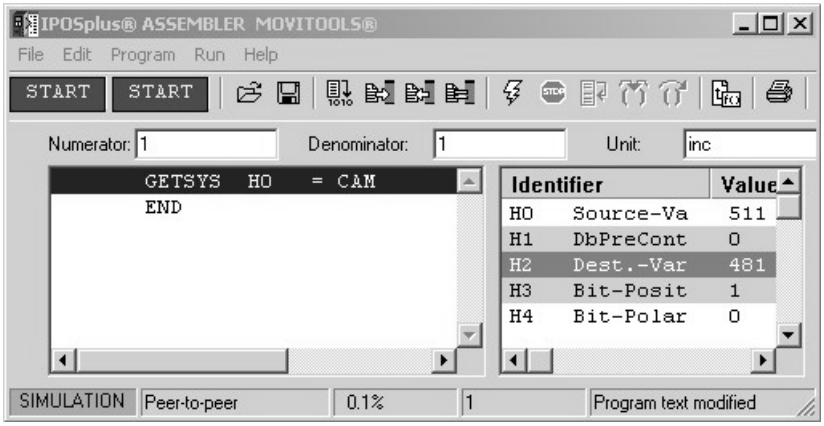
O comando GETSYS carrega o valor de um sistema interno do argumento X2 para uma ou mais variáveis do argumento X2.

Estrutura do comando

Estrutura do comando Mxxx GETSYS X1 X2	Mxxx:	Label (opcional)
	X1:	Hxxx = Inicia a estrutura da variável que contém o resultado após o comando ter sido realizado.

X2:	
ACTUAL CURRENT	Corrente ativa em 0.1 % da corrente nominal do conversor
ACT.SPEED	Rotação atual em 0.1 % rpm
SETP.SPEED	Rotação de ajuste em 0.1 % rpm
ERROR	Código da irregularidade de acordo com a tabela "Mensagens de irregularidade e listas de irregularidade" no manual de sistema
SYSTEM STATUS	Estado de operação, valor do display de 7 segmentos sem estado de irregularidade de acordo com a tabela "Indicação do modo de operação" no manual de sistema
ACT.POSITION	Posição atual dependendo do encoder selecionado em P941 (H509, H510 ou H511)
SETP.POSITION	Posição de ajuste (seleção atual do setpoint do gerador de perfil durante a realização de um comando de deslocamento), idêntico a variável do sistema H491
TARGET POSITION	Posição final, idêntico à variável do sistema H492
INPUTS	Entradas digitais H483 (MOVIDRIVE® A) / H520 (MOVIDRIVE® B) da unidade básica e opcionais; idêntico a variável do sistema H483.
DEVICE STATUS	Idêntico à palavra de estado 1 do protocolo fieldbus (código da irregularidade + estado de operação)
OUTPUTS	Saídas digitais H482 (MOVIDRIVE® A) / H521 (MOVIDRIVE® B) da unidade básica e opcional
IxT	Utilização em 0.1 % da corrente nominal do conversor
ACT.POSITION / SETP.POSITION / TARGET POSITION: A resolução depende do encoder selecionado em P941: – Encoder do motor: 4096 Inc./volta – Encoder externo X14: Resolução do encoder P944 – DIP (Encoder SSI): Resolução do encoder P955	

ANALOG INPUTS	–10 V ... 0 ... +10 V = –10000 ... 0 ... 10000	
	H+0	Valor de tensão da entrada analógica 1 [mV]
	H+1	Valor de tensão da entrada analógica 2 [mV]
CAM		O comando **GETSYS H = CAM** representa um controlador came. Utilizando o comando GETSYS, pode ser utilizado um controlador came padrão com 1 saída por came, para cada conversor. Com o novo conversor MOVIDRIVE® (MDx_A como da versão .14, MCH como da versão .13 e MDx_B), pode ser utilizado um controlador came expandido com 8 saídas. Hxx é a primeira variável de uma estrutura de dados (CamControl). O bit com maior valor (bit 31) é utilizado em Hxx para decidir qual controlador came o comando GETSYS refere-se. Bit 31 = 0: Controlador came padrão (todos os conversores MOVIDRIVE®). O comando GETSYS ativa o controlador came. Os cames são formados uma vez quando o comando GETSYS é processado. Se o controlador came padrão deve processar ciclicamente, o comando deve ser acessado ciclicamente. Bit 31 = 1: Controlador came expandido (somente MDx_A como da versão .14, MCH como da versão .13 e MDx_B com versão aplicação e modo de operação CFC ou SERVO). O comando GETSYS ativa o controlador came, os cames são formados ciclicamente na base. A estrutura das variáveis depende se é acessado o controlador came padrão ou expandido. A estrutura de dados é descrita na seção "Detecção da posição e Posicionamento / controladores came"
ANALOG OUTPUTS	+/– 10 V corresponde a +/– 10000	
	H	A variável no comando GETSYS H = ANALOG OUTPUTS define o início da estrutura da variável seguinte.
	H+0	Contém o valor de tensão da saída analógica 1 (AO1)
	H+1	Contém o valor de tensão da saída analógica 1 (AO2)
TIMER 0		Carrega o valor atual do timer 0 [ms], idêntico a variável do sistema H489
TIMER 1		Carrega o valor atual do timer 1 [ms], idêntico a variável do sistema H488
PO-DATA		Leitura do buffer de dados PO (dados enviados do mestre para o conversor). Os 3 ítens de dados PO ou os 10 ítens de dados PO são lidos dependendo do número de ítens de dados PO.
	H+0	Tipo de rede 0 = Reservado 1 = TERMINAL 2 = RS-485 3 = Fieldbus 4 = Reservado 5 = SBus 8 = SBus 2 (somente MOVIDRIVE® B)
	H+1	Número de dados PO
	H+2	PO1
	H+3	PO2
	H+4	PO3
	H+5	PO4
	H+6	PO5
	H+7	PO6
	H+8	PO7
	H+9	PO8
	H+10	PO9
	H+11	PO10
DC-VOLTAGE		Tensão do circuito intermediário [V]



RELATED TORQUE **REL. TORQUE VFC**		Torque relativo / torque relativo VFC. O torque relativo é o valor indicado baseado na corrente nominal do conversor para o torque no eixo de saída do motor. O torque absoluto pode ser calculado a partir deste valor utilizando a seguinte fórmula: $M_{abs} = M_{rel} \times I_N \times M_N / 1000 / I_{QN}$ M_{abs} = Torque absoluto I_N = Corrente nominal do conversor M_{rel} = Torque relativo baseado em 0.1 % I_N M_N = Torque nominal do motor [Nm] I_{QN} = Corrente nominal Q [A] para o tipo de conexão selecionado. O valor é disponível nos modos de operação CFC e SERVO / VFC1, VFC1 & hoist, VFC1 & DC braking e VFC1 & flying start.
ACT. SPEED EXT.		Rotação atual do encoder externo X14. É utilizada a seguinte estrutura de dados:
	H	Time Base 5 ms ... 31 ms: Valor médio para detecção da rotação do encoder externo.
	H+1	Tipo do encoder 0 = Encoder X14 1 = Encoder DIP
	H+2	Numerator -2^{15} ... 0 ... + 2^{15} –1: Numerador para escala do usuário.
	H+3	Denominator 1 ... 2^{15} –1: Denominador para escala do usuário.
	H+4	D-Pointer 0 ... 458: Indicador para resultar a variável H'.
	H'	Result unit: [nX14] = Inc./time base.

	Exemplo: A rotação do encoder mestre deve ser indicada em voltas por hora. Com um filtro de valor médio de 30 ms e encoder X14, as voltas por hora são calculadas conforme a seguir:
	11250 / 384 = (1000 ms × 60 s × 60 min) / (Incr. por rotação em plena carga × base do tempo)
	O sinal negativo em H32 faz com que o sentido de rotação seja invertido.
	`SET H30 = 30`
	`SET H31 = 0`
	`SET H32 = -11250`
	`SET H33 = 384`
	`SET H34 = 40`
	`GETSYS H30 = ACT.SPEED EXT.`	

SPEED MON. TIMER		Valor numerador da função do controle de rotação. O comando GETSYS pode ser utilizado como um pré-aviso para o controle de rotação. O controle de rotação é ativado quando a corrente estiver no limite atual para o número de segundos especificados em P501. Por exemplo, se P501 = 200 ms, o valor numerador pode ser verificado com GETSYS. Deste modo, o deslocamento de retorno pode ser feito em velocidade rápida e, quando sob carga, a rotação pode ser reduzida automaticamente pelo conversor.

Ajuste dos comandos variable SET / fault response SETFR / Indirect addressing SETI / Interrupt SETINT / system values SETSYS

SET

O comando SET carrega o argumento X1 com o conteúdo do argumento X2 (variável H ou constante K). O resultado é escrito no argumento X1, o argumento X2 permanece inalterado.

Estrutura do comando

Estrutura do comando Mxxx SET X1 = X2	Mxxx:	Label (opcional)
	X1:	Hxxx = Resultado do comando.
	X2:	Hyyy = Fonte.

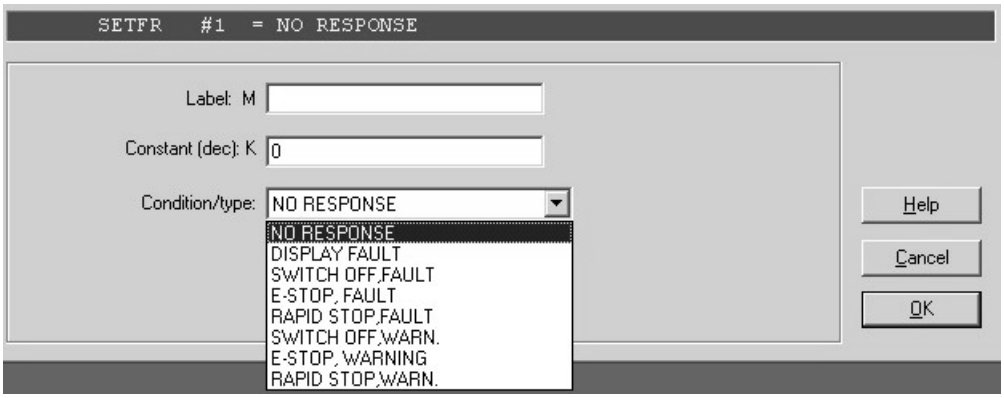
SETFR / SET FAULT REACTION

O comando SETFR define a reação a uma falha do conversor. O código da falha é inserido no argumento X1 do comando. A reação à falha do conversor é selecionada com o argumento 2. A reação da falha selecionada, somente será realizada depois do comando SETFR ter sido processado. A reação da falha selecionada **mais recente** (acesso do comando SETFR ou mudanças em P83_ "Fault response") será a reação a ser indicada.

Pode-se programar todas as reações à uma irregularidade contanto que tenha um ponto na coluna "P" na lista de irregularidades nas instruções de operação ou no manual de sistema.

Exemplo:

- Fault 27 "Limit switches missing", nenhum ponto na coluna P: Não pode ser programada
- Fault 28 "Fieldbus timeout", ponto na coluna P: programável



Estrutura do comando

Estrutura do comando Mxxx SETFR #X1 = X2	Mxxx:	Label (opcional)
	X1:	Código da falha a qual a reação deve se aplicar. São permitidos os seguintes números de erro: 08: n-monitoring 11: Overtemperature 26: External terminal (P830) 28: Fieldbus timeout (P831) 31: TF sensor (P835) 39: Reference travel 42: Lag error (P834) 43: RS-485 timeout (P833) 47: SBus timeout (P836) 77: IPOS control word 78: IPOS software limit switch (P838) 84: Motor protection (P832) 92: DIP operating range 93: DIP absolute encoder

X2:	
NO RESPONSE	**Sem reação** (e sem indicação de falha).
DISPLAY FAULT	**Sem reação, somente a falha é indicada** (o nível do borne de uma saída programada para "/FAULT" é ajustado de 1 até 0).
SWITCH OFF, FAULT	**O estágio de saída é bloqueado, sem torque, o freio é ativado.** Após o reset: Reação quanto a energização/desenergização: O programa IPOS$^{plus®}$, posição de referência, saídas, parâmetros (comando SETSYS) e variáveis ajustadas pelo IPOS$^{plus®}$ são reativados (o programa inicia na linha 1).
E-STOP, FAULT	**O conversor é parado na rampa de parada de emergência.** Após o reset: Reação → SWITCH OFF, FAULT.
RAPID STOP, FAULT	**O conversor é parado na rampa de parada rápida.** Após o reset: Reação → SWITCH OFF, FAULT.
SWITCH OFF WARNING	**O estágio de saída é bloqueado, sem torque, o freio é ativado.** O programa IPOS$^{plus®}$ continua, posição de referência, saídas, parâmetros (comando SETSYS) e variáveis ajustadas pelo IPOS$^{plus®}$ são retidos.*
E-STOP, WARNING	**O conversor é parado na rampa de parada de emergência.** O programa IPOS$^{plus®}$ mantém em funcionamento, → SWITCH OFF, WARNING.*
RAPID STOP, WARNING	**O conversor é parado na rampa de parada rápida.** O programa IPOS$^{plus®}$ mantém em funcionamento, → SWITCH OFF, WARNING.*
* mesmo após confirmação de falha	

SETI / SET INDIRECT H = [H]

A variável X1 recebe o valor da variável cujo número está contido na variável X2.

Para o número das variáveis endereçadas indiretamente fora da faixa definida (por ex. MOVIDRIVE® A faixa 0 ... 512), é gerada a mensagem de falha IPOS INDEX OVERFL (32).

Estrutura do comando

Estrutura do comando Mxxx SETI X1 = [X2]	Mxxx:	Label (opcional)
	X1:	Hxxx = Variável final.
	X2:	Hyyy = Número da variável fonte.

Exemplo

```
SET H1 = 7
SET H7 = 11
SET H3 [H1]
```

Após o programa funcionar, as variáveis têm os seguintes valores:

H1 = 7

H7 = 11

H3 = 11

SETI / SET INDIRECT [H] = H

A variável com o número na variável X1 recebe o valor da variável X2.

Para o número das variáveis endereçadas indiretamente fora da faixa definida (por ex. MOVIDRIVE® A faixa 0 ... 512), é gerada a mensagem de falha IPOS INDEX OVERFL (32).

Estrutura do comando

Estrutura do comando Mxxx SETI [X1] = X2	Mxxx:	Label (opcional)
	X1:	Hxxx = Número da variável final.
	X2:	Hyyy = Variável fonte.

Exemplo

```
     SET   H01  = 50
     SET   H0   = 10
M01 :SETI  [H0] = H01
     ADD   H0   + 1
     ADD   H01  + 10
     JMP   H0   <= 15     , M01
```

Após o programa funcionar, as variáveis têm os seguintes valores:

H10 = 50

H11 = 60

...

H15 = 100

SETINT / SET INTERRUPT

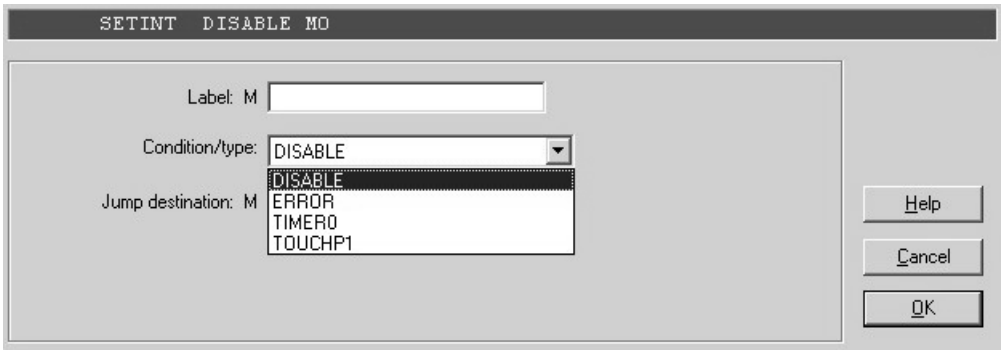
O comando SETINT ajusta o endereço de partida de uma rotina de interrupção. O endereço é indicado como label no comando. Uma interrupção pode ser provocada por vários acontecimentos. Os acontecimentos são especificados no Argumento X1. A rotina de interrupção deve ser completada com um comando RET.

Um salto para a rotina de interrupção ocorre imediatamente e não depende da linha do programa principal processada atualmente. Se a rotina de interrupção termina com o comando RET, o processamento do programa continua do ponto onde a interrupção ocorreu (o processamento de uma interrupção "wait command" é continuado).

O comando SETINT somente está ativo no task 1 e o processamento do task 1 é interrompido durante o processamento da interrupção.

Somente é possível processar uma interrupção por vez, embora uma interrupção com prioridade mais alta possa interromper o processamento de outra interrupção. O ERROR tem prioridade mais alta, depois TOUCH PROBE, seguido por TIMER 0.

Uma interrupção tem que ser inicializada somente uma vez utilizando SETINT.



Estrutura do comando

Estrutura do comando Mxxx SETINT X1, Myyy	Mxxx: Label (opcional)

X1:	
DISABLE	Desativando a interrupção, o indicador de salto (Mxx) é sem importância.
ERROR	Ativa uma interrupção no caso de uma falha do conversor. A rotina de interrupção funciona ciclicamente até o erro ser retirado, no ponto, que a rotina é deixada utilizando o comando RET. Dependendo da reação de irregularidade ajustada (**grupo de parâmetro 830** ou comando **SETFR**), o processamento da rotina de interrupção resultará em um comportamento diferente do descrito acima: • Nenhuma interrupção é realizada se as falhas no grupo de parâmetro 830 forem ajustadas para "No response" ou se o comando SETFR for ajustado para "NO RESPONSE". • O programa é reiniciado (ver o comando SETFR) após reconhecimento da falha, se a reação à irregularidade (**grupo de parâmetro 830** ou o comando **SETFR**) for ajustada para "..., **FAULT**". Qualquer referência ID que foi ajustada é perdida.
TIMER 0	Ativa uma interrupção quando o ajuste de tempo no Timer 0 H489 ocorre. Um "auto reload" com a variável do sistema H485 ocorre após o Timer 0 ter ocorrido. Este valor de recarga determina o tempo com o qual a rotina de interrupção é realizada ciclicamente.
TOUCH PROBE	Ativa uma interrupção quando há uma mudança do nível de sinal no terminal touch probe DI02, se o touch probe foi ativado para o terminal DI02 (parâmetro P601 = IPOS INPUT) e o comando TOUCHP foi transmitido.

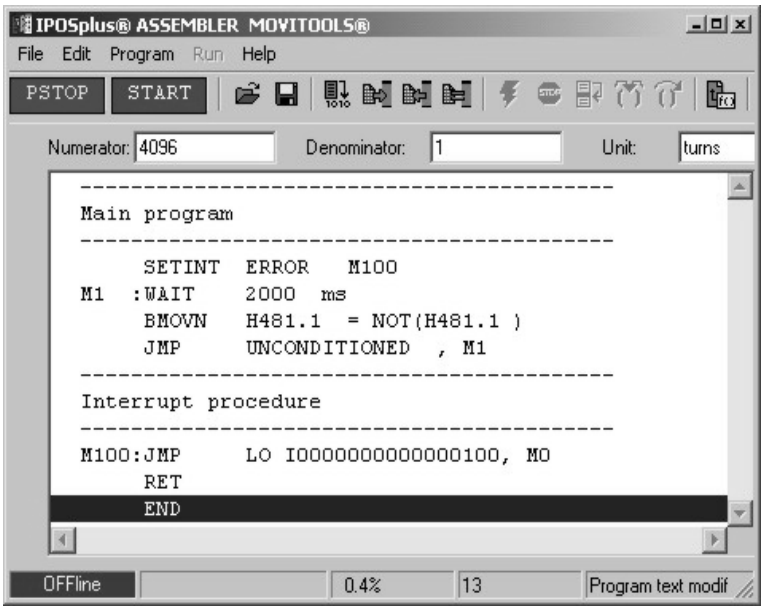
	Myyy: Inicia a label da rotina de interrupção.

Exemplo 1

Rotina de interrupção no caso de uma irregularidade do conversor

No exemplo de programa, a saída digital DO01 é alternada após uma pausa de 2 seg. O programa desvia imediatamente para a rotina de interrupção se ocorrer uma irregularidade do conversor. O sistema volta (RET) para o programa principal assim que houver um sinal "high" no terminal DI02. O parâmetro para entrada DI02 deve ser ajustado para "Reset" para reativar a falha.

Os erros da unidade MQX podem ajustar a interrupção para ERROR. Os erros da unidade do MOVIMOT® conectado não podem provocar a interrupção.



Exemplo 2

O exemplo a seguir ilustra o princípio deste processo:

```
M0  :SETINT  ERROR   M01
     JMP     UNCONDITIONED   , M0
M01 :ADD     H0 + 1
     RET
     END
```

H0 faz a contagem contanto que o erro do conversor ainda ocorra. Após reativar a irregularidade, H0 contém o valor da EEPROM, por exemplo, 0. O valor da memória de trabalho calculado durante o erro é perdido.

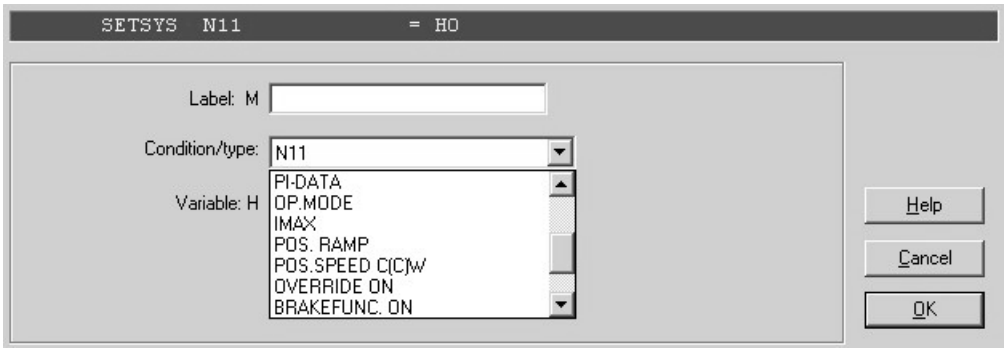
SETSYS

O comando SETSYS escreve o valor de uma ou mais variáveis ao valor do sistema interno. O primeiro argumento seleciona o valor do sistema a ser escrito enquanto o segundo argumento contém o número da (primeira) variável fonte.

Os valores do sistema são reativados aos seus valores originais quando o sistema é desligado (rede e alimentação 24 V).

Observe

Os valores do sistema de escrita podem alterar os ajustes do conversor que foram feitos para a aplicação durante a colocação em operação. Em particular, as mudanças das rampas de posicionamento e a corrente máxima devem ser adaptadas às características do sistema para eliminar os riscos e danos (por ex. devido a sobrecarga mecânica).



Estrutura do comando

Estrutura do comando Mxxx SETSYS X1, X2	Mxxx:	Label (opcional)
	X2:	Número da primeira variável fonte.

X1: Valores do sistema que podem ser selecionados		
	O setpoint fixo interno (grupo de parâmetro P160/ P170) pode ser alterado em etapas de 0.1 rpm utilizando o programa IPOS$^{plus®}$ (mesmo durante o deslocamento se não houver regulador bloqueado). Importante: O novo ajuste fixo somente é adotado após 5 ms. Pode-se atrasar o processamento do programa após um comando SetSys com o comando de espera (5 ms).	
N11 = N12 = N13 = N21 = N22 = N23 =	Setpoint fixo interno n11 Setpoint fixo interno n12 Setpoint fixo interno n13 Setpoint fixo interno n21 Setpoint fixo interno n22 Setpoint fixo interno n23	
PI DATA	Os dados de entrada de processo correspondem ao protocolo fieldbus	
	H H+1 H+2 H+3	Número de ítens de dados PI Dado PI 1 Dado PI 2 Dado PI 3
OP. MODE	Ajuste do modo de operação: O modo de operação somente pode ser mudado dentro do mesmo procedimento de controle (CFC ou SERVO) (mesmo durante o deslocamento, se não houver regulador bloqueado).	
	11 12 13 14 16 17 18 19	CFC (controle de rotação) CFC & torque control CFC & IPOS (posicionamento) CFC & synchronous operation (DRS11A) SERVO (controle de rotação) SERVO & torque control SERVO & IPOS (posicionamento) SERVO & synchronous operation (DRS11A)
IMAX	Ajuste da corrente máxima (somente jogo de parâmetro 1) como uma porcentagem da corrente nominal do conversor (faixa de ajuste: 0.1 ... 150 %, em etapas de 0.1 %); os ajustes também podem ser feitos durante o deslocamento.	

POS. RAMP	Rampas de posicionamento (aceleração/desaceleração); os ajustes também podem ser feitos durante o deslocamento (somente para o tipo de rampa "linear"). Ajuste em ms com valor de referência 3000 rpm. H Rampa de posicionamento 1 (aceleração) H+1 Rampa de posicionamento 2 (desaceleração)
POS. SPEED	Rotação de posicionamento (Horária/Antihorária); os ajustes também podem ser feitos durante o deslocamento (somente para o tipo de rampa "linear"). Ajuste em 0.1 rpm. H Rotação de posicionamento Horária H+1 Rotação de posicionamento Antihorária
OVERRIDE ON	Liga/desliga o override; os ajustes também podem ser feitos durante o deslocamento (somente para o tipo de rampa "linear"). H = 0 Desl H = 1 Lig
BRAKE FUNC. ON	Liga/desliga a função freio H = 0 Desl H = 1 Lig
RAMP TYPE	Não devem ser feitos ajustes durante o deslocamento (torque shocks!). Mudança P916) H = 0 linear H = 1 senoidal H = 2 quadrática H = 3 rampa via rede H = 4 Suavização de picos H = 5 Came eletrônico H = 6 I-Operação em sincronismo
RESET ERROR	Reativa o erro do sistema na variável X2
ACT. POSI-TION	Ajusta a posição atual do encoder do motor ACTPOS.MOT (H511)
SPLINE MULTIAXIS	Cálculo interno do conversor de um cam disc analítico. Atualmente, a função somente é disponível em MCH na versão SD disponível 0C.
	O cálculo spline é inicializado através da função do sistema após até 20 pontos da curva (x-y = valor em pares, x = posição mestre, y = posição escravo) terem sido especificados em uma faixa de encoder mestre. O cálculo é então iniciado utilizando `h+0 SplineMode` e um cam disk completo ou um segmento de um cam disk selecionado ser preenchido. Atualmente, é disponível um procedimento spline 0 (para funcionamento otimizado) e um procedimento spline 1 (para movimentos seção-por-seção e seções diretas). O cálculo é completo após ≤ 200 ms. – H+0 = SplineMode: (Faixa de valor: 0 ... 3) • = 0: Interpolação não ativa, ou cálculo é terminado • = 1: Iniciar interpolação, inserir valores interpolados do index 0 iniciando com o came eletrônico (em ordem crescente; isto é, do index 0 até 512). • = 2: Iniciar interpolação, inserir valores interpolados do index 512 iniciando com o came eletrônico (em ordem decrescente; isto é, do index 512 até 0). • = 3: Cálculo do parâmetro preparatório para interpolação concluída; iniciar inserindo valores interpolados no came eletrônico. – H+1 = SplineModeControl: Reservado – H+2 = SplineDest: (Faixa de valor: 0 ... 5) Número do came eletrônico o qual os valores interpolados devem ser inseridos. – H+3 = SplineNUser: (Faixa de valor: 2 ... 10) Número dos pontos da curva a ser utilizado para interpolação e o processo de cálculo (bit 7 = 0 spline 0, bit 7 = 1 spline 1) – H+4 = SplineX0User: (Somente um valor >= 0 pode ser inserido aqui!) Valor X do primeiro ponto da curva – H+5 = SplineY0User: (Faixa de valor: distância = -2^{31} ... 0 ... (2^{31} –1)) Valor Y (= valor de posição) do primeiro ponto da curva – ... – H+42 = SplineX19User: (Somente um valor <= 512 pode ser inserido aqui!) Valor X do 20º ponto da curva – H+43 = SplineY19User: (Faixa de valor: distância = -2^{31} ... 0 ... (2^{31} –1)) Valor Y do 20º ponto da curva
	SS_MULTIAXIS: Cálculo total de uma trajetória. Disponível somente sob consulta. Ver também o adendo para as instruções de operação 'Special design SK-0C for MCH: Calculated Curves with MCH.“.

_SetVarInterrupt

Syntax

```
VarInt Hxx, Mxx
```

Descrição

Este comando não está disponível no MOVIDRIVE® A, somente no MOVIDRIVE® B.

O comando ativa uma interrupção da variável com a estrutura de dados como da variável Hxx. Se a condição para a interrupção for realizada e o task 2 ou 3, a qual esta interrupção é processada, for iniciado, os comandos são realizados como da label Mxx. O evento para a interrupção é a comparação com um valor da variável (ver H+4). Se a estrutura de dados foi inicializada durante o funcionamento, o comportamento da interrupção pode ser adaptado dinamicamente ao VarInterrupt completo, utilizando um comando IPOS$^{plus®}$.

Observe: O dado da estrutura de dados somente é transferido quando o comando `VarInt Hxx, Mxx` é acessado (consistência de dados).

Exemplo: Se o valor da estrutura de dados Hx+3 CompareVar é mudado, por exemplo, o valor somente é considerado com o comando `VarInt Hxx, Mxx`.

Argumentos

Hxx Primeira variável de uma estrutura de dados (ver tabela H+0)

Mxx Label com o primeiro comando da função interrupção.

Estrutura de dados da interrupção da variável:

Variável	Estrutura do elemento VARINT	Descrição
H+0	Control	0: Todas VarInterrupt = OFF / Reset 1: A interrupção recebe o tempo de processamento do task 2 e interrompe este task enquanto a interrupção é processada. 2: A interrupção recebe o tempo de processamento do task 3 e interrompe este task enquanto a interrupção é processada.
H+1	IntNum	0 ... 3: Define um número seqüencial da VarInterrupt. Uma interrupção com o número x, que já tenha sido ativada, pode ser reativada durante o tempo de funcionamento do programa com outra estrutura de dados, utilizando o acesso do comando `VarInt Hxx, Mxx` quando o mesmo número de interrupção é especificado na nova estrutura de dados na posição H+1. Esta característica não é disponível para as interrupções do task 1.
H+2	SrcVar	Número da variável de referência de quem o valor é comparado com o valor de comparação. ScrVar é o valor da variável de referência que ScrVar refere-se.
H+3	CompVar	Valor de comparação ou máscara utilizada para comparar o valor da variável de referência H+2.
H+4	Mode	0: Sem ocorrência de interrupção. Pode ser utilizado para desativar esta interrupção sem desativar todas. 1: Um dos bits da variável de referência, exibido utilizando a máscara CompVar, mudou seu estado: ([*SrcVar(t) ^ *SrcVar(t-T)] & CompVar) != 0 2:Contanto que o valor da variável de referência seja igual ao valor de comparação (*SrcVar == CompVar) 3:Contanto que o valor da variável de referência não seja igual ao valor de comparação (*SrcVar != CompVar) 4:Contanto que o valor da variável de referência seja maior ou igual ao valor de comparação (*SrcVar >= CompVar) 5:Contanto que o valor da variável de referência seja menor ou igual ao valor de comparação (*SrcVar <= CompVar) 6: Valor da variável de referência E o valor de comparação não são 0 ((*SrcVar & CompVar) != 0) 7: Valor da variável de referência E o valor de comparação são 0 ((*SrcVar & CompVar) == 0) 8: Pulso positivo do bit exibido pelo CompVar 9: Pulso negativo do bit exibido pelo CompVar 10: Como 2; no entanto, a interrupção somente é processada cada vez que a condição é realizada (pulso ativado) 11: Como 3; no entanto, a interrupção somente é processada cada vez que a condição é realizada (pulso ativado) 12: Como 4; no entanto, a interrupção somente é processada cada vez que a condição é realizada (pulso ativado) 13: Como 5; no entanto, a interrupção somente é processada cada vez que a condição é realizada (pulso ativado)
H+5	Priority	Prioridade da interrupção (1 ... 10); task 2 e task 3 são atribuídas a prioridade 0.
H+6	IntEvent	Imagem do processo da variável de referência de *SrcVar até o tempo da interrupção.

Exemplo

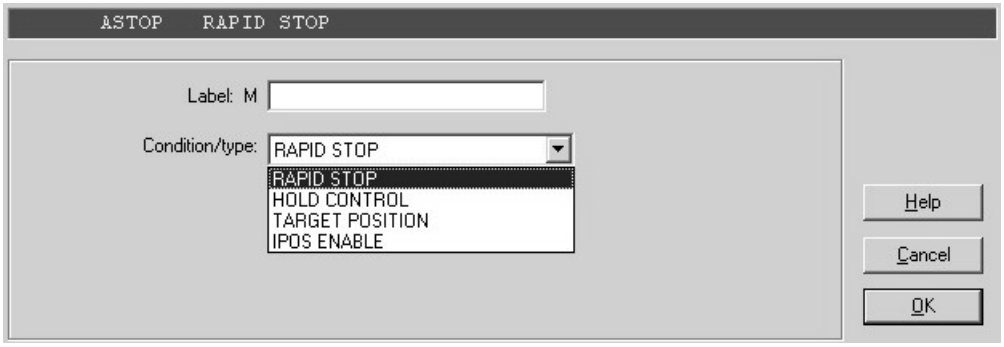
Ver "Task Management and Interrupts / Variable interrupts with MOVIDRIVE® B".

20.8 Comandos especiais

ASTOP / MEM / TOUCHP / WDOFF / WDON

ASTOP / AXIS STOP

O comando ASTOP é utilizado para parar ou liberar o conversor (ver H484 bit 1). O argumento do comando (RAPID STOP, HOLD CONTROL, TARGET POSITION) define o tipo de parada (rampa, controle quando parado, etc.) ou libera o conversor (IPOS ENABLE).



Estrutura do comando

Estrutura do comando Mxxx ASTOP X1	Mxxx: Label (opcional)

X1:	
RAPID STOP	Frenagem com a rampa de parada rápida seguida pelo controle de rotação. A última posição final (H492) que foi transmitida é retida. O bloqueio através da palavra de controle (comando ASTOP (IPOS ENABLE) é necessário antes do comando de deslocamento posterior). O freio é atuado se a função do freio estiver ativada.
HOLD CONTROL	Frenagem com a rampa da unidade básica (P131/P133) seguida pelo controle de posição. A última posição final (H412) que foi transmitida é retida. O bloqueio através da palavra de controle (o comando ASTOP (IPOS ENABLE) é necessário com o comando de deslocamento posterior). O freio não é atuado se a função do freio estiver ativada.
TARGET POSITION	Parada de posicionamento com a rampa de posicionamento (P911 / P912) e calculada a posição final "STOP" (possível somente no modo de posicionamento), seguida pelo controle de posição. A última posição final (H492) que foi transmitida é regravada pela posição de parada. Nenhum bloqueio através da palavra de controle (nenhum comando ASTOP (IPOS ENABLE) necessário antes do comando de deslocamento posterior). O freio não é atuado se a função do freio estiver ativada. Observe: Desde que a posição atual seja utilizada como a posição do valor nominal em parada, o comando não pode ser processado ciclicamente. Este é o caso em eixos com processo de força ou elevação porque caso contrário o eixo desvia da posição lentamente.
IPOS ENABLE	O bloqueio é cancelado utilizando a palavra de controle IPOS$^{plus®}$.

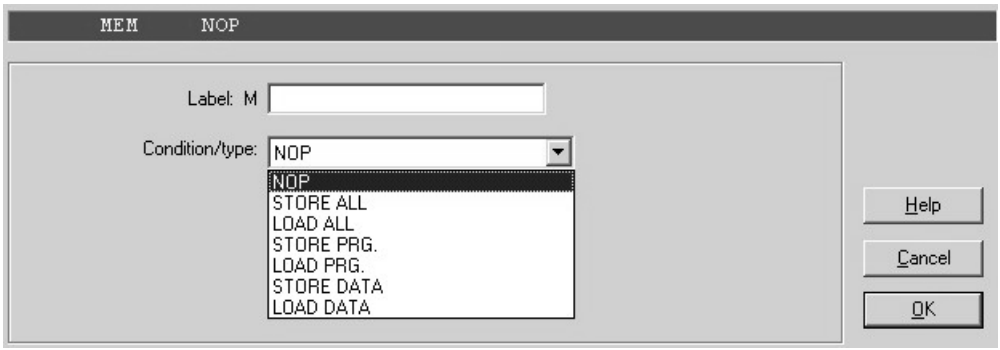
MEM / MEMORIZE

O comando MEM possibilita salvar (carregar) os programas IPOS$^{plus®}$ e/ou as variáveis na (da) memória não-volátil no (para o) conversor. A ação é especificada através do argumento.

Observe

Ao utilizar o comando MEM observe que as variáveis armazenadas na memória não-volátil (H0 127) e todos os parâmetros não são escritos ciclicamente. Isto é porque o número de operações de armazenagem com a média de armazenamento (EEPROM) é restrito a 10^5 operações.

As variáveis individuais também podem ser armazenadas utilizando o comando MOVILINK.



Estrutura do comando

Estrutura do comando Mxxx MEM X1	Mxxx: Label (opcional)

X1:	
NOP	Nenhum dado é armazenado
STORE ALL	Os programas e os dados na memória de trabalho são salvos na memória não-volátil (EEPROM)
LOAD ALL	Os programas e os dados são carregados da memória não-volátil (EEPROM) para a memória de trabalho.
STORE PROG.	Somente o programa da memória de trabalho é salvo na memória não-volátil (EEPROM)
LOAD PROG.	Somente o programa da memória não-volátil (EEPROM) é carregado na memória de trabalho
STORE DATA	Somente as variáveis da memória de trabalho são salvas na memória não-volátil (EEPROM)
LOAD DATA	Somente as variáveis da memória não-volátil são carregadas (EEPROM) na memória de trabalho

Exemplo

Se ocorrer um erro, o programa salta para uma rotina de interrupção de erro. Aqui, o comando MEM STORE DATA é acessado para que você possa continuar o processamento com o estado provisório armazenado das variáveis H0 ... H127 após a restauração do erro.

Caso não utilize este comando, o programa inicia com os valores mais recentes da EEPROM e regrava os valores de trabalho mais recentes.

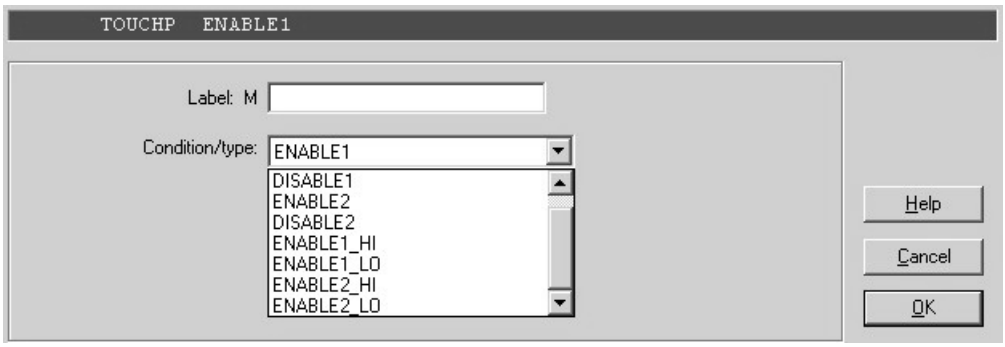
TOUCHP / TOUCH PROBE

O comando **TOUCHP** libera ou bloqueia uma entrada touch probe. A função touch probe geralmente é atribuída aos terminais de entrada DI02 e / ou DI03. As entradas utilizadas para a função touch probe devem ser ajustadas para "IPOS input" para evitar que sejam alocadas duas vezes.

Se houver uma mudança do nível de sinal na entrada touch probe após o comando TOUCHP ter sido realizado, as posições atuais de corrente (H511, H510, H509) são armazenadas uma vez nas variáveis destinadas à esta finalidade (H502 H507).

Um contador somente é disponível em MQX e MOVITRAC® 07 com a variável H511. Para tomar outra medida, o touch probe deve ser novamente liberado.

As posições do touch probe são armazenadas dentro de 100 µs, independente do processamento do programa existente. O nível do terminal deve ser alterado para que pelo menos 200 µs sejam seguramente detectados. O argumento pode ser utilizado para selecionar a mudança do pulso que produz uma touch probe.



As posições touch probe são armazenadas nas seguintes variáveis:			
Encoder	Posição do encoder	Posição Touch probe 1 (DI02)	Posição Touch probe 2 (DI03)
Encoder do motor (X15)	H511 ACTPOS. MOT	H507 TP.POS1MOT	H505 TP.POS2MOT
Encoder externo (X14)	H510 ACTPOS.EXT	H506 TP.POS1EXT	H504 TP.POS2EXT
Encoder absoluto (X62)	H509 ACTPOS.ABS	H503 TP.POS1ABS	H502 TP.POS2ABS
Encoder virtual (somente para MOVIDRIVE® B)	H376	H501 TpPos1_VE	H500 TpPos2_VE

Estrutura do comando

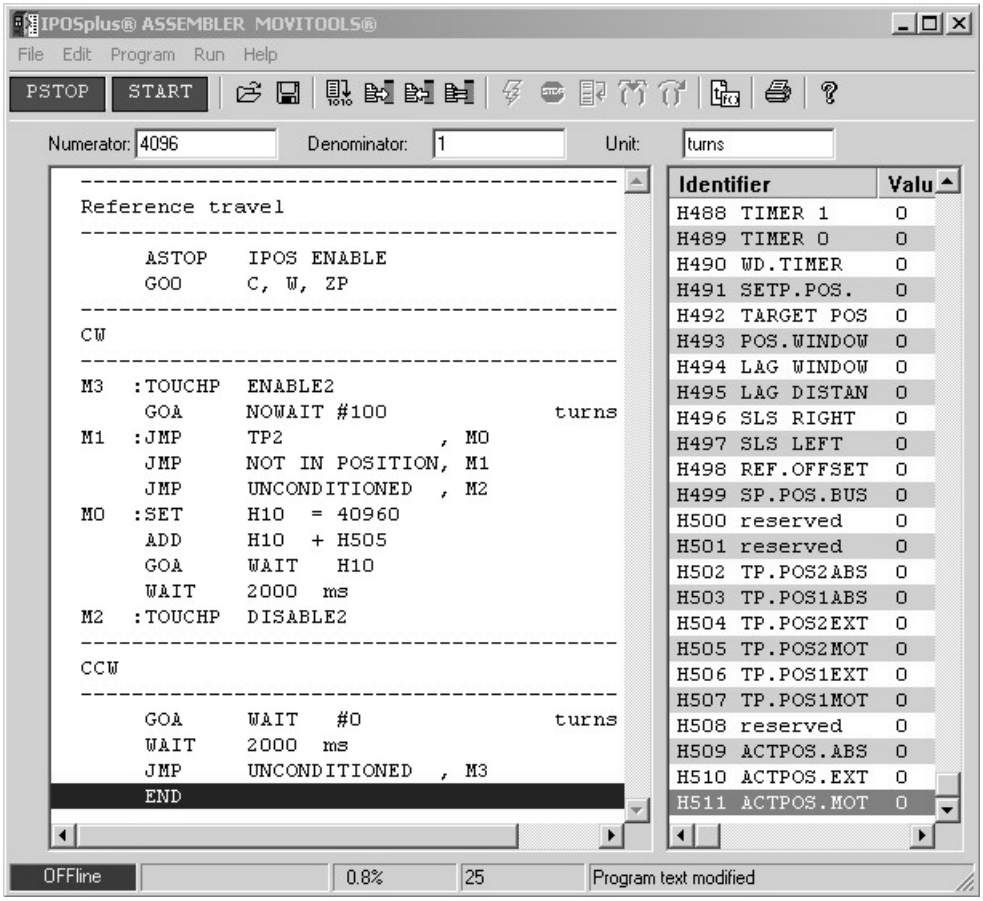
Estrutura do comando Mxxx TOUCHP X1	Mxxx: Label (opcional)

X1:	
ENABLE 1	Permite a entrada da touch probe DI02. Quando o sinal muda negativo/positivo e positivo/negativo, as posições atuais são armazenadas.
DISABLE 1	Bloqueia a entrada da touch probe DI02
ENABLE 2	Permite a entrada da touch probe DI03. Quando o sinal muda negativo/positivo e positivo/negativo, as posições atuais são armazenadas.
DISABLE 2	Bloqueia a entrada da touch probe DI03
ENABLE 1_HI	Permite a entrada da touch probe DI02. Quando o sinal muda negativo/positivo, as posições atuais são armazenadas.
ENABLE 1_LO	Permite a entrada da touch probe DI02. Quando o sinal muda positivo/negativo, as posições atuais são armazenadas.
ENABLE 2_HI	Permite a entrada da touch probe DI03. Quando o sinal muda negativo/positivo, as posições atuais são armazenadas.
ENABLE 2_LO	Permite a entrada da touch probe DI03. Quando o sinal muda positivo/negativo, as posições atuais são armazenadas.

O usuário pode determinar se uma entrada touch probe foi ativada no programa, por exemplo, com MP TP2, M0 ou com SETINT TOUCHP1 M0. O usuário pode determinar se um valor de posição armazenado encontra-se em uma faixa de posição específica comparando os valores com os do programa do usuário a seguir.

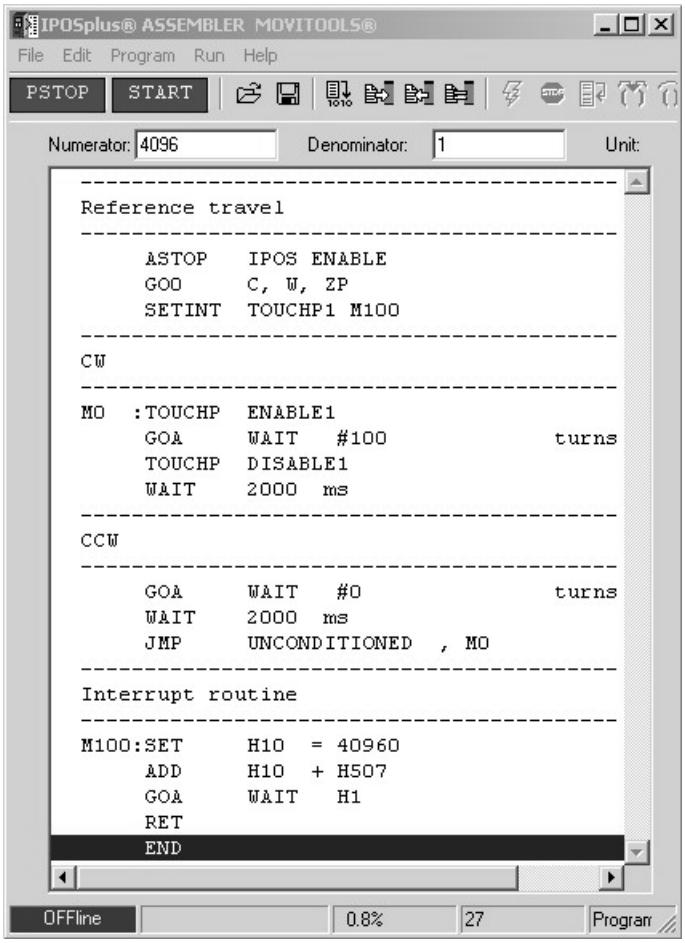
Exemplo 1

No programa, o conversor desloca entre as posições absolutas 0 voltas e 100 voltas. Se houver uma mudança de pulso na entrada touch probe DI03 enquanto o conversor movimenta-se para a posição final de 100 voltas, são deslocadas precisamente mais 10 voltas (40960 incr.) desta posição touch probe. Para voltar à posição 0, a função touch probe é desativada utilizando o comando DISABLE2.



Exemplo 2

Como alternativa ao exemplo acima, pode ser realizado um desvio do programa (indicador de salto M100) quando a posição touch probe é alcançada. Isto é obtido utilizando o comando "SETINT TOUCHP1 M100".



WDOFF / WDON / WATCH DOG OFF / ON

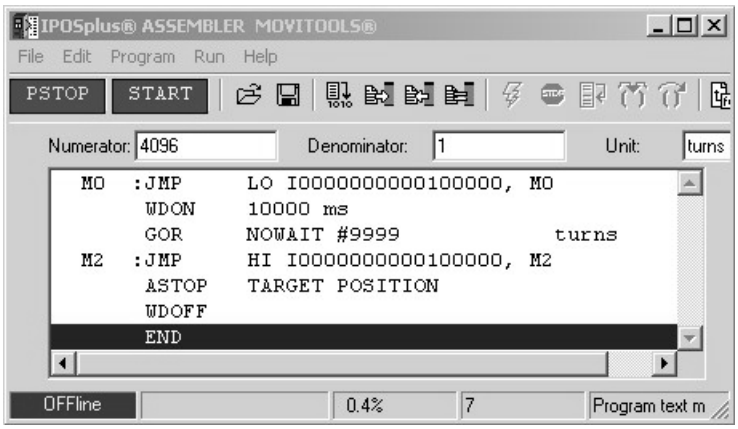
O watchdog é acessado no intervalo de tempo especificado no argumento. Todos os tasks são interrompidos e o conversor é parado com a falha 41 se o tempo especificado no timer watchdog H490 ocorrer antes da função de monitoração ser desligada utilizando o comando WDOFF. (O estágio de saída é bloqueado e o freio é atuado. Se não houver freio, o conversor roda através da inércia até parar.)

Estrutura do comando

Estrutura do comando		
Mxxx WDON X1 Mxxx WDOFF	Mxxx:	Label (opcional)
	X1:	Intervalo em ms o qual o watchdog é acessado.

Exemplo

O conversor movimenta-se enquanto o nível em DI05 é ajustado para 1 ("positivo"). A função "watchdog" garante que o conversor não desloca por mais do que 10 seg. Caso seja excedido o limite de 10 segundos, o conversor é parado.



20.9 Comandos de comparação

Operações de comparação CPEQ /CPGE /CPGT / CPLE / CPLET /CPNE

Uma variável é comparada com um 2º argumento (variável ou constante). São possíveis as seguintes comparações:

- Igual a (CPEQ)
- Maior ou igual a (CPGE)
- Maior do que (CPGT)
- Menor ou igual a (CPLE)
- Menor do que (CPLT)
- Diferente de (CPNE)

Além disso o resultado pode ser processado com um comando de salto posterior.

CPEQ / COMPARE EQUAL

O comando CPEQ compara, observando os sinais, se a variável X1 é a mesma da variável ou constante X2. A variável X1 contém o resultado. É diferente de zero se a condição for realizada; caso contrário, o resultado é zero.

Além disso o resultado pode ser processado, por exemplo, com um comando de salto posterior. A variável X2 permanece inalterada.

Estrutura do comando

Estrutura do comando Mxxx CPEQ X1 == X2		
	Mxxx:	Label (opcional)
	X1:	Variável (resultado)
	X2:	Variável ou constante

Exemplo 1

```
SET  H0  = 13
SET  H1  = 50
CPEQ H0  == H1
```

Após o programa ser processado, H0 tem valor zero e H1 tem valor 50.

Exemplo 2

```
SET  H0  = 13
CPEQ H0  == 13
```

Após o programa ser processado, H0 tem valor um.

CPGE / COMPARE GREATER OR EQUAL

O comando CPGE compara, observando os sinais, se a variável X1 é maior ou igual à variável ou constante X2. A variável X1 contém o resultado. É diferente de zero se a condição for realizada; caso contrário, o resultado é zero.

Além disso o resultado pode ser processado, por exemplo, com um comando de salto posterior. A variável X2 permanece inalterada.

Estrutura do comando

Estrutura do comando Mxxx CPGE X1 >= X2		
	Mxxx:	Label (opcional)
	X1:	Variável (resultado)
	X2:	Variável ou constante

Exemplo 1

```
SET  H0  = 13
SET  H1  = 50
CPGE H0  >= H1
```

Após o programa ser processado, H0 tem valor zero e H1 tem valor 50.

Exemplo 2

```
SET  H0  = -3
CPGE H0  >= -3
```

Após o programa ser processado, H0 tem valor um.

CPGT / COMPARE GREATER THAN

O comando CPGT compara, observando os sinais, se a variável X1 é maior do que a variável ou constante X2. A variável X1 contém o resultado. É diferente de zero se a condição for realizada; caso contrário, o resultado é zero.

Além disso o resultado pode ser processado, por exemplo, com um comando de salto posterior. A variável X2 permanece inalterada.

Estrutura do comando

Estrutura do comando Mxxx CPGT X1 > X2	Mxxx:	Label (opcional)
	X1:	Variável (resultado)
	X2:	Variável ou constante

Exemplo 1

```
SET  H0  = -3
CPGT H0  > -3
```

Após o programa ser processado, H0 tem valor zero.

Exemplo 2

```
SET  H0  = 3
SET  H2  = 2
CPGT H0  > H2
```

Após o programa ser processado, H0 tem valor um.

CPLE / COMPARE LESS OR EQUAL

O comando CPLE compara, observando os sinais, se a variável X1 é menor ou igual à variável ou constante X2. A variável X1 contém o resultado. É diferente de zero se a condição for realizada; caso contrário, o resultado é zero.

Além disso o resultado pode ser processado, por exemplo, com um comando de salto posterior. A variável X2 permanece inalterada.

Estrutura do comando

Estrutura do comando Mxxx CPLE X1 <= X2	Mxxx:	Label (opcional)
	X1:	Variável (resultado)
	X2:	Variável ou constante

Exemplo 1

```
SET  H0  = 50
SET  H1  = 13
CPLE H0  <= H1
```

Após o programa ser processado, H0 tem valor zero e H1 tem valor 13.

Exemplo 2

```
SET  H0  = -3
CPLE H0  <= -3
```

Após o programa ser processado, H0 tem valor um.

CPLT / COMPARE LESS THAN

O comando CPLT compara, observando os sinais, se a variável X1 é menor do que a variável ou constante X2. A variável X1 contém o resultado. É diferente de zero se a condição for realizada; caso contrário, o resultado é zero.

Além disso o resultado pode ser processado, por exemplo, com um comando de salto posterior. A variável X2 permanece inalterada.

Estrutura do comando

Estrutura do comando Mxxx CPLT X1 < X2	Mxxx:	Label (opcional)
	X1:	Variável (resultado)
	X2:	Variável ou constante

Exemplo 1

```
SET  H0  = -3
CPLT H0  < -3
```

Após o programa ser processado, H0 tem valor zero.

Exemplo 2

```
SET  H0  = 2
SET  H2  = 3
CPLT H0  > H2
```

Após o programa ser processado, H0 tem valor um.

CPNE / COMPARE NOT EQUAL

O comando CPNE compara, observando os sinais, se a variável X1 é diferente da variável ou constante X2. A variável X1 contém o resultado. É diferente de zero se a condição for realizada; caso contrário, o resultado é zero.

Além disso o resultado pode ser processado, por exemplo, com um comando de salto posterior. A variável X2 permanece inalterada.

Estrutura do comando

Estrutura do comando Mxxx CPNE X1 != X2	Mxxx:	Label (opcional)
	X1:	Variável (resultado)
	X2:	Variável ou constante

Exemplo 1

```
SET  H0  = 13
SET  H1  = 13
CPNE H0  != H1
```

Após o programa ser processado, H0 tem valor zero e H1 tem valor 13.

Exemplo 2

```
SET  H0  = 50
CPNE H0  == 13
```

Após o programa ser processado, H0 tem valor um.

Operações lógicas ANDL / ORL / NOTL

ANDL / LOGICAL AND

O comando ANDL é a operação lógica AND de duas variáveis. O resultado é escrito à variável X1. A variável X2 permanece inalterada. O resultado é zero quando uma das duas variáveis = 0. O resultado é um, quando ambas variáveis != 0.

Estrutura do comando

Estrutura do comando Mxxx ANDL X1 && X2	Mxxx:	Label (opcional)
	X1:	Variável (resultado)
	X2:	Variável

Exemplo 1

```
SET  H01  = 100
SET  H02  = 0
ANDL H01  && H02
```

Após o programa ser processado, H01 tem valor zero.

Exemplo 2

```
SET  H01  = 100
SET  H02  = 50
ANDL H01  && H02
```

Após o programa ser processado, H01 tem valor um.

ORL / LOGICAL OR

O comando ORL é a operação lógica OR de duas variáveis. O resultado é escrito à variável X1. A variável X2 permanece inalterada. O resultado é um, quando uma das duas variáveis != 0. O resultado é zero quando ambas variáveis = 0.

Estrutura do comando

Estrutura do comando Mxxx ORL X1 || X2	Mxxx:	Label (opcional)
	X1:	Variável (resultado)
	X2:	Variável

Exemplo 1

```
SET  H01  = 100
SET  H02  = 0
ORL  H01  || H02
```

Após o programa ser processado, H01 tem valor um.

Exemplo 2

```
SET  H01  = 0
SET  H02  = 0
ORL H01  || H02
```

Após o programa ser processado, H01 tem valor zero.

NOTL / LOGICAL NOT

O comando NOTL cumpre a negação lógica de uma variável. O resultado é escrito à variável X1. A variável X2 permanece inalterada. O resultado é um, quando a variável X2 = 0. O resultado é zero quando a variável X2 != 0.

Estrutura do comando

Estrutura do comando Mxxx NOTL X1 = NOT (X2)		
	Mxxx:	Label (opcional)
	X1:	Variável (resultado)
	X2:	Variável

Exemplo 1

```
SET   H02  = 100
NOTL  H01  NOT (H02)
```

Após o programa ser processado, H01 tem valor zero.

Exemplo 2

```
SET   H02  = 0
NOTL  H01  NOT (H02)
```

Após o programa ser processado, H01 tem valor um.

21 Assembler – Exemplos

21.1 Exemplo do programa "Flashing light"

Exemplo "controle"

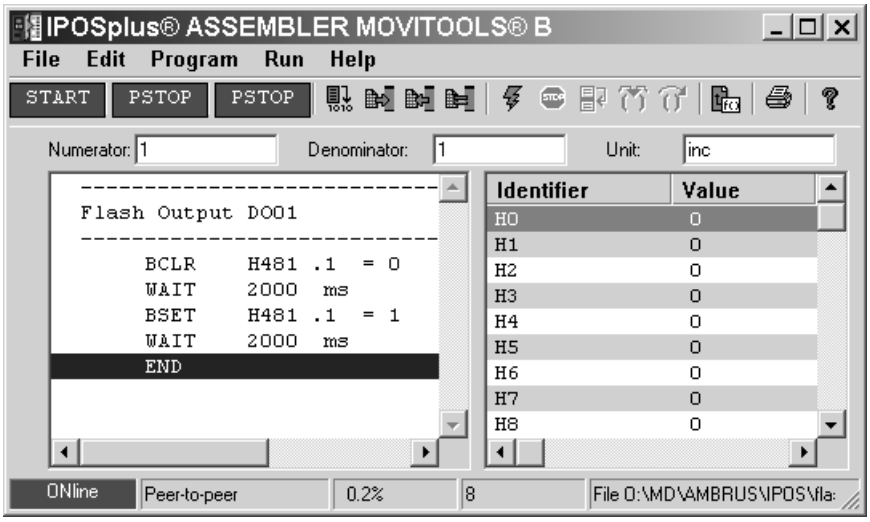
Este exemplo do programa liga e desliga a saída digital DOØ1 a cada 2 segundos.

Partida rápida (exemplo)

Pré-requisitos

Ligação à rede e/ou 24 V (terminais de alimentação lógica X10:9 (+24 V / VI24) e X10:10 (0 V / DGND)) conectada; sem necessidade de conectar o motor e o encoder (nenhum movimento do motor).

1. Não há necessidade de colocar em operação o controle de rotação.
2. Ajustar os parâmetros para a saída em Shell (P621 Binary output DOØ1 → IPOS OUTPUT).
3. Iniciar o assembler a partir do MOVITOOLS® Manager.
4. Abrir/ativar a janela "Program" e inserir o exemplo do programa "Flash output DOØ1."
5. Efetuar o download do exemplo do programa a partir da janela do programa (PC) para a memória de programa do conversor: Pressionar "Ctrl + F9" na janela ativa do programa
6. Iniciar o exemplo do programa: Pressionar "F9" na janela ativa do programa.
7. Verificar o programa do usuário:
   - O task 1 indica no cabeçalho do programa as mudanças de PSTOP para START.
   - O indicador do programa funciona na janela do programa.
   - Em Shell, a indicação do parâmetro P052, o borne de saída DOØ1, mudam entre 1 e 0 a cada 2 segundos.



10340AXX

Figura 77: Janela do programa Assembler

O programa consiste de:
3 linhas de comentário (duas linhas tracejadas para destacar o nome do programa, e uma linha para o nome do programa)
Ajuste da saída DO02 (X13:3) para "0"
Espera de 2 segundos
Ajuste da saída DO02 (X13:3) para "1"
Espera de 2 segundos
Fim do programa / salta para o início do programa

Exemplo "Posicionamento"

Este exemplo do programa alterna a posição do conversor 10 voltas do motor sentido horário e antihorário a cada 2 s.

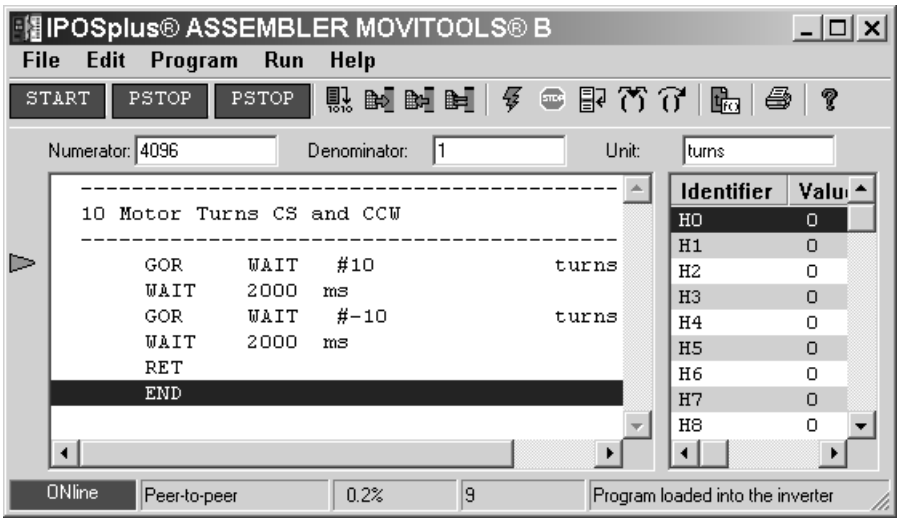
Partida rápida (exemplo)

Pré-requisitos

- São conectados conversor / motor / encoder
- Conversor colocado em operação de acordo com o manual de sistema MOVIDRIVE® no modo de operação VFC-n-CTRL & IPOS, CFC & IPOS ou SERVO & IPOS (P700). P700 deve ter um dos ajustes após a colocação em operação.
- Verificar as chaves fim-de-curso do hardware do circuito EMERGENCY OFF.

1. Ajuste do parâmetro:
   - P600 Binary input DI01 → ENABLE / STOP
   - P601 Binary input DI02 → NO FUNCTION
   - P602 Binary input DI03 → NO FUNCTION
   - P603 Binary input DI04 → /LIM. SWITCH CW
   - P604 Binary input DI05 → /LIM. SWITCH CCW
   - P700 Operating mode → (VFC-n-Reg. / CFC / SERVO) & IPOS
   - NUMERATOR → 4096
   - DENOMINATOR → 1
   - UNIT → Rev.
2. Inserir o exemplo do programa "10 motor revolutions forwards and backwards".
3. Efetuar o download do exemplo do programa: Pressionar "F2" na janela ativa do programa.
4. O conversor não deve alcançar as chaves fim-de-curso. Os terminais DIØ4 (X13:5) e DIØ5 (X13:6) devem ter o nível "1".
5. Iniciar o exemplo do programa: Pressionar "F9" na janela ativa do programa.
6. Verificar o exemplo do programa:
   - O task 1 indica no cabeçalho do programa as mudanças de PSTOP para START.
   - O motor movimenta alternadamente 10 voltas sentido horário ou antihorário a cada dois segundos.
   - A mudança na posição pode ser sinalizada na indicação do parâmetro P003.

- O valor do ajuste de posição e a posição atual são indicados nas variáveis H492 e H511.



10341AXX

Figura 78: Exemplo do programa "10 motor revolutions forwards and backwards"

O programa consiste de:
3 linhas de comentário
Deslocamento relativo 10 voltas do motor sentido horário
Espera de 2 segundos
Deslocamento relativo 10 voltas do motor sentido antihorário
Espera de 2 segundos
Fim do programa
Fim do programa / salta para o início do programa

O comando RET não é obrigatório neste exemplo porque o programa não foi acessado como uma subrotina. O comando de retorno faz com que o programa volte para a primeira linha, que é permitido neste caso.

21.2 Exemplo do programa "hoist"

Características

- Referenciamento
- Seleção de três posições de elevação através das entradas digitais
- Notificação quando é alcançada uma posição selecionada
- Movimento automático distante das chaves fim-de-curso do hardware

O conversor pode mover-se 3 posições utilizando os 3 primeiros bornes de entrada do opcional DIO11B.

O conversor é movido distante da chave fim-de-curso do hardware por meio de um sinal "1" na entrada "RESET" (DI02).

Ajustes

Está disponível uma descrição detalhada da configuração das entradas/saídas na seção de comentário do código fonte do programa.

Estrutura esquemática

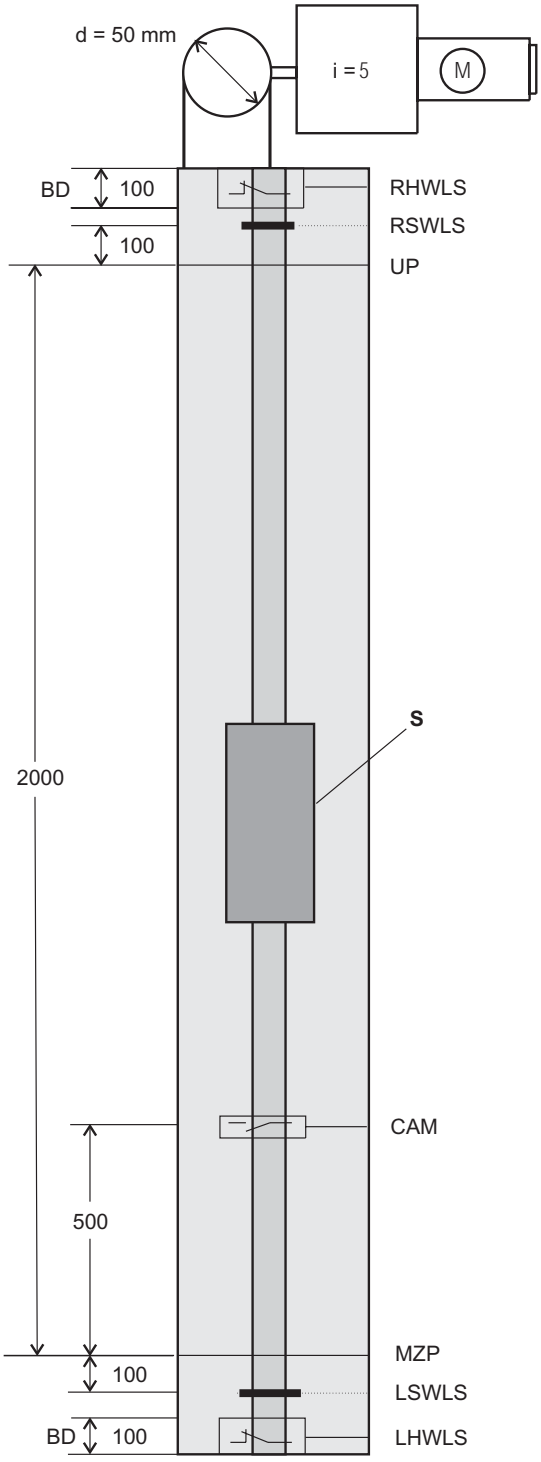
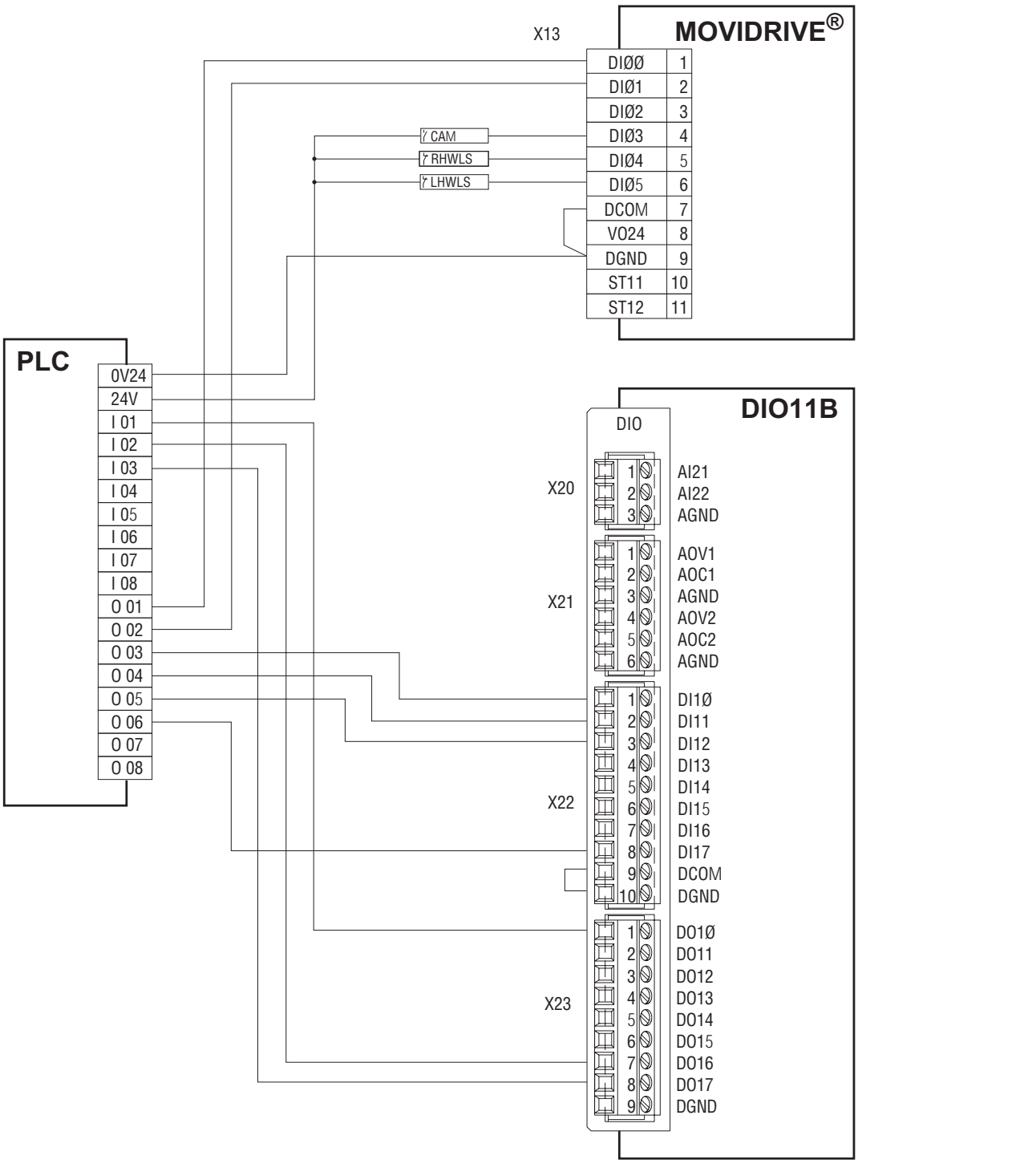


Figura 79: Estrutura esquemática de elevação com IPOS^{plus}®

BD =	Distância de frenagem	CAM =	Came de referência
RHWLS =	Chave fim-de-curso do hardware sentido horário	MZP =	Zero da máquina
RSWLS =	Chave fim-de-curso do software sentido horário	LSWLS =	Chave fim-de-curso do software sentido antihorário
UP =	Faixa de deslocamento superior	LHWLS =	Chave fim-de-curso do hardware sentido antihorário
S =	Carro de deslocamento		

Ligação dos bornes



55219AXX

Figura 80: Esquema de ligação IPOS$^{plus®}$

PLC = Controlador externo
DI00 = /Regulador bloqueado
DI01 = Liberação/Parada rápida
DI02 = Reset
DI03 = Came de referência
DI04 = /Chave fim-de-curso horária
DI05 = /Chave fim-de-curso antihorária
DI10 = Posição 1
DI11 = Posição 2
DI12 = Posição 3
DI16 = Início do referenciamento
DI17 = Início do posicionamento
DO10 =/Irregularidade
DO16 =IPOS em posição
DO17 =Referência IPOS

Ajuste dos parâmetros relacionado com o exemplo

Grupo	Parâmetros	Ajuste
30_Limits	P302 Maximum speed 1 [rpm] P350 Change direction of rotation	1500 OFF
60_ Binary inputs basic unit	P600 Binary input DI01 P601 Binary input DI02 P602 Binary input DI03 P603 Binary input DI04 P604 Binary input DI05	ENABLE / STOP RESET REFERENCE CAM /LIM. SWITCH CW /LIM. SWITCH CCW
61_Binary inputs option DIO11A	P610 Binary input DI10 ... P617 Binary input DI17	IPOS INPUT ... IPOS INPUT
63_Binary outputs option DIO11A	P630 Binary output DO10 P636 Binary output DO16 P637 Binary output DO17	/FAULT IPOS IN POSITION IPOS REFERENCE
7__Control functions	P700 Operating mode P730 Brake function	CFC & IPOS YES
9__ IPOS parameters	P900 Reference offset [mm] P901 Reference speed 1 [rpm] P902 Reference speed 2 [rpm] P903 Reference travel type P910 Gain X controller P911 Positioning ramp 1 [s] P912 Positioning ramp 2 [s] P913 Travel speed CW [rpm] P914 Travel speed CCW [rpm] P915 Speed feedforward [%] P916 Ramp type P920 SW limit switch CW [mm] P921 SW limit switch CCW [mm] P922 Position window [inc] P923 Lag error window [inc] P930 Override	500 200 50 1 2.8 1 1 1350 1350 100 SINE 2100 –100 50 5000 OFF
Travel distance factors NUMERATOR/DENOMINATOR	Travel distance factor NUMERATOR Travel distance factor DENOMINATOR Unit	2048000 15708 mm

Cálculo dos parâmetros IPOS^{plus}®

Chave fim-de-curso de software
Ver estrutura esquemática

Distância de deslocamento fator numerador
A unidade de dimensão do deslocamento deve ser ajustada para mm!

Número de incrementos por volta da roda propulsora

Incr./ volta do motor × redução

4096 incrementos × 5 = 20480

20480 × 100 (fator de extensão) =2048000

Distância de deslocamento fator denominador
Circunferência da roda propulsora em mm

d × π

50 mm × π = 157,0796327

157.08 × 100 (fator de extensão) =15708

Conversor
O conversor após a informação específica de deslocamento deve aparecer em mm.

Velocidade de deslocamento
1350 rpm

Janela de posição
A mensagem 'Drive in position' deve ser emitida quando é alcançada a posição final ±50 incrementos.

Terminais de entrada

Nível	Terminal	Função do terminal	Descrição
0	DI00	/Regulador bloqueado	Liga/desliga o módulo de potência
0	DI01	Liberação	Parada controlada
0	DI02	Reset	Reset após irregularidade (alívio da chave fim-de-curso)
0	DI03	Came de referência	Comuta para posição zero ou valor de offset
0	DI04	Chave fim-de-curso horária	Chave fim-de-curso para parada (+)
0	DI05	Chave fim-de-curso antihorária	Chave fim-de-curso para parada (–)
0	DI10	Entrada IPOS	Posição de elevação 0 mm
0	DI11	Entrada IPOS	Posição de elevação 1000 mm
0	DI12	Entrada IPOS	Posição de elevação 2000 mm
0	DI13	Entrada IPOS	–
0	DI14	Entrada IPOS	Jog positivo
0	DI15	Entrada IPOS	Jog negativo
0	DI16	Entrada IPOS	Início do referenciamento
0	DI17	Entrada IPOS	Início do posicionamento

Terminais de saída

Nível	Terminal	Unidade	Função do terminal	Descrição
0	DB00	MDX	/Freio	Ativação do freio através do relé auxiliar
0	DO01	MDX	Pronto	Controlador ativo; fonte de alimentação à eletrônica OK
0	DO02	MDX	/Irregularidade	Nenhuma irregularidade presente
0	DO10	DIO11B	Saída IPOS	–
0	DO11	DIO11B	Saída IPOS	–
0	DO12	DIO11B	Saída IPOS	–
0	DO13	DIO11B	Saída IPOS	–
0	DO14	DIO11B	Saída IPOS	–
0	DO15	DIO11B	Saída IPOS	–
0	DO16	DIO11B	IPOS em posição	O conversor está na janela de posição
0	DO17	DIO11B	Referência IPOS	Referenciamento realizado com sucesso

Código fonte do programa (com comentários)

NUMER.: 2048000 DENOM. 15708 UNIT: mm ********** Program: Hoist With the first 3 inputs of the option DIO11A, the drive moves to position 0;1000;2000. File: Hub 100.mdx Author: SEW/AWT Date: 01.06.98 Modified: 01.06.98 Terminal wiring of inputs:---------- DI00 Controller inhibit DI01 Enable DI02 Reset (move clear of LS) DI03 Reference cam DI04 = CW limit switch DI05 CCW limit switch DI10 Hoist position 0 mm DI11 " 1000 mm DI12 " 2000 mm DI13 - - - DI14 (Jog CW) DI15 (Jog CCW) DI16 Reference travel DI17 Start positioning Terminal wiring of outputs:---------- DB00 Brake DO01 Ready signal DO16 "IPOS in position" DO17 "IPOS reference"	Comentário
---------- Program start ========== Program branch distributor ========== SETINT ERROR M10 M100:CALL M50 JMP LO I0001000000000000, M101 CALL M20 M101:JMP LO I0000010000000000, M102 CALL M30 M102:JMP LO I0000100000000000, M103 CALL M40 M103:JMP UNCONDITIONED , M100 ----------	Distribuidor de desvio do programa Ativa a rotina de interrupção para processamento da chave fim-de-curso do hardware Reset/alívio da chave fim-de-curso →Programa principal DI16 = 1 → Referenciamento DI15 = 1 → Jog Hor. DI14 = 1 → Jog Antih.
Reset subroutine/move clear of limit switch ========== M10:JMP HI I0000000000110000, M1 M3: JMP HI I0000000000110000, M2 ASTOP IPOS ENABLE JMP UNCONDITIONED , M3 M2: ASTOP TARGET POSITION M1 :RET ----------	Reset/alívio da chave fim-de-curso Se não houver contato com a chave fim-de-curso (DI04/DI05 Limit switch CW/CCW), voltar ao distribuidor de desvio. Se houver, então destrave o deslocamento e espere até o conversor ter aliviado a chave fim-de-curso (DI02 input terminal função "Reset") Depois páre o conversor ajustando a posição final para a posição atual
Subroutine: Referencing ========== M20:ASTOP IPOS ENABLE GO0 U,NW, ZP M22:JMP LO I0000000000000001, M21 SET H319 = 0 BMOV H319.0 = H473.20 JMP H319 == 0 , M22 M21 : ASTOP TARGET POSITION RET ----------	Referenciamento Destravar o deslocamento Referenciamento, sem espera, inicia no pulso zero, Cancela o referenciamento e o bit na palavra de estado "IPOS Reference" = 0
(Subroutine: Jog mode) ========== M30:RET M40:RET ----------	Opção: Subrotina (por ex. operação manual) Jogging Hor. Jog Antih. Ver exemplo a seguir.
Main program: Hoist positioning ========== M50 :JMP LO I0000000001000000, M51 GOA WAIT #0 mm M51:JMP LO I0000000010000000, M52 GOA WAIT #1000 mm M52:JMP LO I0000000100000000, M53 GOA WAIT #2000 mm M53:RET ---------- END	Programa principal: Posicionamento de elevação Se a entrada DI10 é ajustada, mover para a posição 0 mm Se a entrada DI11 é ajustada, mover para a posição 1000 mm Se a entrada DI12 é ajustada, mover para a posição 2000 mm

21.3 Exemplo do programa "Jog mode"

Características

- Deslocamento em dois sentidos com entradas digitais Jog+ / Jog–.
- Rotações e rampas de deslocamento ajustáveis.
- Deslocamento de um sentido possível.
- Nenhum referenciamento necessário.
- Conformidade com os limites da faixa de deslocamento (chaves fim-de-curso do software).
- Movimento automático distante das chaves fim-de-curso do hardware.

O movimento de um sentido é possível nos dois sentidos utilizando duas entradas digitais Jog+ (DI14) e Jog– (DI15). Nenhum referenciamento é necessário. Se o conversor tiver sido referenciado e as chaves fim-de-curso do software ajustadas, o deslocamento ocorre somente dentro destes limites. O movimento ocorre somente quando há sinal "1" em um dos terminais jog. O conversor movimenta-se distante da chave fim-de-curso do hardware inserindo sinal "1" na entrada RESET (DI02).

Ajustes

A configuração detalhada das entradas/saídas e das variáveis utilizadas no programa é documentada na seção comentário do código fonte do programa.

Terminais de entrada

Nível	Terminal	Função do terminal	Descrição
0	DI00	/Regulador bloqueado	Liga/desliga o módulo de potência
0	DI01	Liberação	Parada controlada
0	DI02	Reset	Reset após irregularidade (alívio da chave fim-de-curso)
0	DI03	Came de referência	Comuta para posição zero ou valor de offset
0	DI04	Chave fim-de-curso horária	Chave fim-de-curso para parada (+)
0	DI05	Chave fim-de-curso antihorária	Chave fim-de-curso para parada (–)
0	DI10	Entrada IPOS	–
0	DI11	Entrada IPOS	–
0	DI12	Entrada IPOS	–
0	DI13	Entrada IPOS	–
0	DI14	Entrada IPOS	Jog positivo
0	DI15	Entrada IPOS	Jog negativo
0	DI16	Entrada IPOS	Início do referenciamento
0	DI17	Entrada IPOS	Início do posicionamento

Terminais de saída

Nível	Terminal	Unidade	Função do terminal	Descrição
0	DB00	MDX	/Freio	Ativação do freio através do relé auxiliar
0	DO01	MDX	Pronto	Controlador ativo; fonte de alimentação à eletrônica OK
0	DO02	MDX	/Irregularidade	Nenhuma irregularidade presente
0	DO10	DIO11B	Saída IPOS	–
0	DO11	DIO11B	Saída IPOS	–
0	DO12	DIO11B	Saída IPOS	–
0	DO13	DIO11B	Saída IPOS	–
0	DO14	DIO11B	Saída IPOS	–
0	DO15	DIO11B	Saída IPOS	–
0	DO16	DIO11B	IPOS em posição	O conversor está na janela de posição
0	DO17	DIO11B	Referência IPOS	Referenciamento realizado com sucesso

Código fonte do programa (com comentários)

NUMER.: 1 DENOM. 1 UNIT: inc ********** Sample program: Jog mode File: Tipp.mdx Author: SEW/AWT Date: 01.06.98 Function: Jog mode Uni-directional travel possible - No axis referencing required - Compliance with travel range limits, software LS - Travel speeds/ramps from H310 - Inputs jog+ (DI14) / jog- (DI15) Parameter setting (P600) of inputs/outputs: In inverter commas = specified function Without inverted commas = IPOS INPUT/OUTPUT Terminal wiring of inputs:---------- DI00 "Controller inhibit" DI01 "Enable" DI02 "Fault reset" (move clear of LS) DI04 "Reference cam" DI03 "CW limit switch" DI05 "CCW limit switch" DI14 Jog CW DI14 Jog CCW DI16 Start reference travel DI17 (Start positioning) Terminal wiring of outputs:---------- DB00 Brake DO01 Ready signal DO16 "IPOS in position" DO17 "IPOS reference" Variables used:---------- H310 = V-jog CW (1/10 rpm) H311 = "CCW" H312 = Acceleration ramp (ms) H313 = deceleration ramp (ms) H316 - H319 = Jog aux. variable ********** ----------	Comentário
Program start ========== Initialization ---------- SET H310 = 5000 SET H311 = 5000 SET H312 = 2000 SET H313 = 2000 ----------	Ajusta os valores de rotação e aceleração para operação manual (ver comentários)
Program branch distributor ========== SETINT ERROR M10 M100:JMP LO I0001000000000000, M101 CALL M20 M101:JMP LO I0000010000000000, M102 CALL M30 M102:JMP LO I0000100000000000, M103 CALL M40 M103:JMP UNCONDITIONED , M100 ----------	Distribuidor de desvio do programa Ativa a rotina de interrupção para processamento da chave fim-de-curso do hardware Reset/alívio da chave fim-de-curso DI16 = 1 → Referenciamento DI15 = 1 → Jog Hor. DI14 = 1 → Jog Antih.
Reset subroutine/move clear of limit switch ========== M10:JMP HI I0000000000110000, M1 M3: JMP HI I0000000000110000, M2 ASTOP IPOS ENABLE JMP UNCONDITIONED , M3 M2: ASTOP TARGET POSITION M1 :RET ----------	Reset/alívio da chave fim-de-curso Se não houver contato com a chave fim-de-curso (DI05/DI06 LS CW/CCW), voltar ao distribuidor de desvio. Se houver, então destrave o deslocamento e espere até o conversor ter aliviado a chave fim-de-curso (parametrizado DI02 input terminal função "Reset"). Depois páre o conversor ajustando a posição final para a posição atual.
Subroutine: Referencing ========== M20:ASTOP IPOS ENABLE GO0 U,NW, ZP M22:JMP LO I0000000000000001, M21 SET H309 = 0 BMOV H309.0 = H473.20 JMP H309 == 0 , M22 M21 :ASTOP TARGET POSITION RET ----------	Referenciamento Destravar o deslocamento Referenciamento, sem espera, inicia no pulso zero contanto que "Regulador bloqueado" = 0 e o bit na palavra de estado "IPOS Reference" = 0

Subroutine: Jog mode ========== Jog mode (software limit switch interrogation active) ---------- M35 :SETSYS POS.SPEED C(C)W = H310 SETSYS POS. RAMP = H312 SET H319 = 0 BMOV H319.0 = H473.20 JMP H319 == 0, M36 SET H319 = H496 OR H319 | H497 JMP H319 == 0, M36 SET H319 = 1 SET H317 = H496 SET H318 = H497 M36 :RET	Operação manual Interrogação da chave fim-de-curso do software ativa Ajuste da rotação Ajuste do tempo de rampa Interrogar: O eixo foi referenciado? (Chave fim-de-curso de software ativa) Interrogar: Ambas faixas do software = 0? (Chave fim-de-curso de software não ativa) Se a chave fim-de-curso está ativa, então ajustar flag H319=1 e carregar as variáveis de deslocamento jog (H317) com as faixas de deslocamento do software (variáveis do sistema H496 e H497)
---------- Jog + ---------- M30:JMP LO I0000010000000000, M31 JMP HI I0000100000000000, M31 CALL M35 JMP H319 == 1 , M32 GETSYS H317 = ACT.POSITION ADD H317 + 4096000 M32 :ASTOP IPOS ENABLE GOA NOWAIT H317 JMP UNCONDITIONED , M30 M31 :ASTOP TARGET POSITION RET ---------- Jog - ---------- M40:JMP LO I0000100000000000, M41 JMP HI I0000010000000000, M41 CALL M35 JMP H319 == 1 , M42 GETSYS H318 = ACT.POSITION SUB H318 - 4096000 M42 :ASTOP IPOS ENABLE GOA NOWAIT H318 JMP UNCONDITIONED , M40 M41 :ASTOP TARGET POSITION RET ---------- End jog mode ---------- END	Jogging Horário Mover enquanto DI14 = 1 e DI15 = 0, Verificar: A chave fim-de-curso do software está ativa? Acrescentar 1000 voltas do motor à posição atual de corrente e movimentar para resultar a nova posição final Jog Antihorário Mover enquanto DI14 = 0 e DI15 = 1, Verificar: A chave fim-de-curso do software está ativa? Acrescentar 1000 voltas do motor à posição atual de corrente e movimentar para resultar a nova posição final

21.4 Exemplo do programa "Table positioning"

Características

- Seleção binária codificada de 16 posições da tabela
- Saída binária codificada da posição da tabela atualmente selecionada
- Sinal definitivo quando é alcançada a posição da tabela selecionada
- Movimento automático distante das chaves fim-de-curso do hardware.

As primeiras 4 entradas digitais do opcional DIO11B podem ser utilizadas selecionando 16 posições da tabela (variáveis de deslocamento H000 ... H015) no formato binário codificado. Quando é selecionado um número variável de deslocamento (indicador da tabela), sempre é representado nas primeiras 4 entradas digitais do DIO11A no formato binário codificado.

O movimento de referência deve ser ativado utilizando entrada DI16 "Reference Travel" antes de poder movimentar o conversor para as posições da tabela. A entrada DI17 "Start positioning" permite o trabalho de deslocamento à posição da tabela ou interrompe-o (no caso do "Controller inhibit" e "Enable" = sinal "1"). Quando é selecionada uma nova posição da tabela, é aconselhável ajustar a entrada DI17 para o sinal "0" até estar certo que todos os bits do indicador da tabela tenham sido ajustados!

O sinal "1" na saída DO15 "Table position valid" indica que foi alcançada a posição selecionada da tabela. Esta saída é restaurada imediatamente quando é selecionada uma nova posição da tabela. Além de avaliar a saída DO16 "IPOS in position", também é possível detectar quando é retirada a posição selecionada da tabela, mesmo quando o controlador está desativado ("Regulador bloqueado" = "0").

O conversor movimenta-se distante da chave fim-de-curso do hardware inserindo sinal "1" na entrada RESET (DI02).

Ajustes

A configuração detalhada das entradas/saídas (ver a seguir) e das variáveis utilizadas no programa é documentada na seção comentário do código fonte do programa.

As posições da tabela podem ser escritas através da janela da variável do assembler ou com o controle manual nas variáveis (H00 ... H15). Isto significa que as variáveis são armazenadas na memória não-volátil.

As unidades de deslocamento do usuário numerador e denominador no cabeçalho da janela de posição não são relevantes aqui porque os valores de posição das variáveis de deslocamento sempre são avaliados em incrementos (4096 incrementos/volta do motor).

Terminais de entrada

Nível	Terminal	Função do terminal	Descrição
0	DI00	/Regulador bloqueado	Liga/desliga o módulo de potência
0	DI01	Liberação	Parada controlada
0	DI02	Reset	Reset após irregularidade (alívio da chave fim-de-curso)
0	DI03	Came de referência	Comuta para posição zero ou valor de offset
0	DI04	Chave fim-de-curso horária	Chave fim-de-curso para parada (+)
0	DI05	Chave fim-de-curso antihorária	Chave fim-de-curso para parada (–)
0	DI10	Entrada IPOS	Indicador da variável bit 2'0
0	DI11	Entrada IPOS	Indicador da variável bit 2'1
0	DI12	Entrada IPOS	Indicador da variável bit 2'2
0	DI13	Entrada IPOS	Indicador da variável bit 2'3
0	DI14	Entrada IPOS	Jog positivo
0	DI15	Entrada IPOS	Jog negativo
0	DI16	Entrada IPOS	Início do referenciamento
0	DI17	Entrada IPOS	Início do posicionamento

Terminais de saída

Nível	Terminal	Unidade	Função do terminal	Descrição
0	DB00	MDX	/Freio	Ativação do freio através do relé auxiliar
0	DO01	MDX	Pronto	Controlador ativo; fonte de alimentação à eletrônica OK
0	DO02	MDX	/Irregularidade	Nenhuma irregularidade presente
0	DO10	DIO11B	Saída IPOS	Indicador da variável bit 2'0
0	DO11	DIO11B	Saída IPOS	Indicador da variável bit 2'1
0	DO12	DIO11B	Saída IPOS	Indicador da variável bit 2'2
0	DO13	DIO11B	Saída IPOS	Indicador da variável bit 2'3
0	DO14	DIO11B	Saída IPOS	–
0	DO15	DIO11B	Saída IPOS	Posição da tabela válida
0	DO16	DIO11B	IPOS em posição	O conversor está na janela de posição
0	DO17	DIO11B	Referência IPOS	Referenciamento realizado com sucesso

Código fonte do programa (com comentários)

NUMER.: 1 DENOM. 1 UNIT: inc ********** Program: Table positioning File: Tab.mdx Author: SEW/AWT Date: 01.06.98 Function: Table positioning: -The first 4 inputs of the DIO11A option are used to select the positions in the corresponding variables 0-15 in binary coded format. -Input DI17 (X22:17) is used to enable the selected travel command Parameter setting of inputs/outputs: In inverter commas = specified function Without inverted commas = IPOS INPUT/OUTPUT Terminal wiring of inputs:---------- DI00 "Controller inhibit" DI01 "Enable" DI02 "Fault reset" (move clear of LS) DI03 "Reference cam" DI04 "CW limit switch" DI05 "CCW limit switch" DI10 Variable pointer bit 2'0 DI11 2'1 DI12 2'2 DI13 2'3 DI14 (Job CW) DI15 (Job CCW) DI16 Start reference travel DI17 Start positioning Terminal wiring of outputs:---------- DB00 Brake DO01 Ready signal DO10 Variable pointer bit 2'0 DO11 2'1 DO12 2'2 DO13 2'3 DO14 - DO15 Table position reached DO16 "IPOS in position" DO17 "IPOS reference" Variables used:---------- H300 = Travel speed CW (1/10 rpm) H301 = Travel speed CCW (1/10 rpm) H302 = Acceleration ramp CW (ms) H303 = Deceleration ramp CCW (linear) H320 - H324 = Auxiliary variables ********** ----------	Comentário
Program start ========== Initialization ---------- SET H300 = 15000 SET H301 = 15000 SET H302 = 1000 SET H303 = 1000 ----------	Ajusta os valores de rotação e aceleração para posicionamento da tabela (ver descrição da variável nos comentários do código fonte do programa)
Program branch distributor ========== SETINT ERROR M10 M100:CALL M50 JMP LO I0001000000000000, M101 CALL M20 M101:JMP LO I0000010000000000, M102 CALL M30 M102:JMP LO I0000100000000000, M103 CALL M40 M103:JMP UNCONDITIONED , M100 ----------	Distribuidor de desvio do programa Ativa a rotina de interrupção para processamento da chave fim-de-curso do hardware Reset/alívio da chave fim-de-curso → Programa principal DI16 = 1 → Referenciamento DI14 = 1 → Jog Hor. DI15 = 1 → Jog Antih.
Reset subroutine/move clear of limit switch ========== M10:JMP HI I0000000000110000, M1 M3: JMP HI I0000000000110000, M2 ASTOP IPOS ENABLE JMP UNCONDITIONED , M3 M2: ASTOP TARGET POSITION M1 :RET ----------	Reset/alívio da chave fim-de-curso Se não houver contato com a chave fim-de-curso (DI04/DI05 Limit switch CW/CCW), voltar ao distribuidor de desvio. Se houver, então destrave o deslocamento e espere até o conversor ter aliviado a chave fim-de-curso (Ajuste do parâmetro "Reset" input função DI02). Depois páre o conversor ajustando a posição final para a posição atual.

Subroutine: Referencing ========== M20:ASTOP IPOS ENABLE AND H480 & FFFFFFF0 hex BCLR H480.5 = 0 GO0 U,NW, ZP M22:JMP LO I0000000000000001, M21 SET H319 = 0 BMOV H319.0 = H473.20 JMP H319 == 0 , M22 M21 :ASTOP TARGET POSITION RET ----------	Referenciamento Destravar o deslocamento Apagar a posição da tabela binária codificada de saída Apagar a saída "Posição válida da tabela" Referenciamento, sem espera, inicia no pulso zero, contanto que "Regulador bloqueado" = 0 e o bit na palavra de estado "IPOS Reference" = 0
Subroutine: Jog mode ========== M30:RET M40:RET	Opção: Subrotina (por ex. operação manual) Jogging Hor. Jog Antih.
Main program: Table positioning ========== Check whether axis is referenced ---------- M50 : SET H321 = 0 BMOV H321.0 = H473.20 JMP H321 != 0, M51 RET ---------- Set travel speed and ramp ---------- M51:SETSYS POS.SPEED C(C)W = H300 SETSYS POS. RAMP = H302 ---------- Read variable pointer into variable H320 ---------- SET H320 = H483 ASHR H320 >> 6 AND H320 & F hex ---------- Check output "Table position reached" ---------- JMP H322 == H320, M54 BCLR H480.5 = 0 M54 :SET H322 = H320 ---------- Output variable pointer in binary coded format ---------- SET H323 = H320 SET H324 = H480 AND H324 & FFFFFFF0 hex OR H323 | H324 SET H480 = H323 ---------- Table positioning enable ---------- M53:JMP LO I0010000000000000, M52 ASTOP IPOS ENABLE GOA NOWAIT [H320] JMP NOT IN POSITION, M53 BSET H480.5 = 1 JMP UNCONDITIONED , M55 ---------- M52:ASTOP HOLD CONTROL M55 :RET ---------- End table positioning control ---------- END	Programa principal: Posicionamento da tabela O movimento para as posições da tabela ocorre somente se o conversor estiver referenciado (DO17 = 10 posição do bit na variável do sistema do terminal de saída H473; parâmetro ajustado para "IPOS reference") Ajustar a velocidade de deslocamento, a rampa de aceleração e de desaceleração Selecionar o indicador da tabela (nr. da variável de deslocamento) no formato binário codificado com 4 entradas (DI10 DI13) Restaurar a saída "Tab. position valid" se o indicador da tabela tiver sido mudado. Armazenar o indicador atual da tabela na variável de comparação. Escrever o indicador da tabela selecionado aos terminais de saída (DO10 DO13) sem alterar outras saídas da variável de saída (H480) Se DI17 = 1, então deslocar para o valor de posição da variável de deslocamento selecionada, caso contrário o conversor pára Restaurar o sinal "Table position selection valid" Retirar a trava de deslocamento Mover para a posição da tabela até a posição ser alcançada ou DI17 = 0. Ajustar o sinal "Table position selection valid" Parar o conversor

22 Índice de Alterações

Foram feitas várias modificações e revisões em todo o manual.

A seguir, uma lista das mudanças que foram feitas nas seções individuais.

Descrição do Sistema	• Dados técnicos: A duração do ciclo para entradas digitais foi completada.
Detecção da Posição e Posicionamento	• Encoder absoluto SSI (DIP) / Colocação em operação para DIP com encoder absoluto / 7. Ajustar fatores do encoder P942 e P943: O gráfico foi mudado. • Referenciamento / Tipo 8: Sem referenciamento.
IPOS$^{plus®}$ e Fieldbus	• Palavras de controle fieldbus e palavras de estado fieldbus / palavra de estado 3: As funções foram revisadas.
Parâmetros IPOS$^{plus®}$	• P90x Referenciamento IPOS / P903 Tipo de referenciamento: Tipo 8 foi acrescentado. • P94x Encoder IPOS / P942 / P943 Encoder factor numerator / denominator: O gráfico foi mudado.
Compiler – Editor	• Manuseio do projeto com MOVIDRIVE® B. • Inserindo comandos: Observação sobre a combinação das teclas Ctrl + Z.
Compiler – Programação	• #define: Observação sobre o mesmo nome.
Compiler – Funções	• Funções padrão / _SBusCommDef / SCD_TRACYCL: A observação foi completada.

Lista de Endereços

Brasil			
Administração **Fábrica** Montadora	**São Paulo**	SEW-EURODRIVE Brasil Ltda. Avenida Amâncio Gaiolli, 50 Rodovia Presidente Dutra, Km 208 Guarulhos - Cep.: 07251-250 SAT - SEW ATENDE - 0800 7700496	SEW SERVICE - Plantão 24 horas Tel. +55 (0) 11 64 89 90 90 Fax +55 (0) 11 64 80 46 18 SEW SERVICE - Horário Comercial Tel. +55 (0) 11 64 89 90 30 www.sew.com.br sew@sew.com.br
Montadora Vendas Service	Santa Catarina Joinville	SEW-EURODRIVE Brasil Ltda. Rua Dona Francisca, 8300 - BL C/MD 7 Distrito Industrial - 89239 970	Tel. +55 (0) 47 30 27 68 86 Fax +55 (0) 47 30 27 68 88 filial.sc@sew.com.br
	Interior de SP Rio Claro	SEW-EURODRIVE Brasil Ltda. Rodovia Washington Luiz, Km72 Condomínio Industrial Compark - 13501-600	Tel. +55 (0) 19 35 22 31 00 Fax +55 (0) 19 35 24 66 53 filial.rc@sew.com.br
Vendas Service	São Paulo	SEW-EURODRIVE Brasil Ltda. Avenida Amâncio Gaiolli, 50 Rodovia Presidente Dutra, Km 208 Guarulhos - Cep.: 07251-250	Tel. +55 (0) 11 64 89 90 00 Fax +55 (0) 11 64 89 90 09 filial.sp@sew.com.br
	Minas Gerais Belo Horizonte	SEW-EURODRIVE Brasil Ltda. Avenida Brigadeiro Eduardo Gomes, 1275 Glória - 30870-100	Tel. +55 (0) 31 21 02 29 05 Fax +55 (0) 31 21 02 29 00 filial.mg@sew.com.br
	Paraná Curitiba	SEW-EURODRIVE Brasil Ltda. Rua Desembargador Westphalen, 3779 Parolin - 80220-031	Tel. +55 (0) 41 3213 58 12 Fax +55 (0) 41 3213 58 00 filial.pr@sew.com.br
	Rio de Janeiro	SEW-EURODRIVE Brasil Ltda. Avenida Evandro Lins e Silva, 840 - Sala 1407 Barra da Tijuca - 22631-470	Tel. +55 (0) 21 21 78 22 21 Fax +55 (0) 21 21 78 22 31 filial.rj@sew.com.br
	Rio Grande do Sul Porto Alegre	SEW-EURODRIVE Brasil Ltda. Avenida Madrid, 168 Navegantes - 90240-560	Tel. +55 (0) 51 30 25 18 25 Fax +55 (0) 51 30 25 18 35 filial.rs@sew.com.br
	Amazonas Manaus	SEW-EURODRIVE Brasil Ltda. Rua Nicolau da Silva, 159 São Francisco - 69063-001	Tel. +55 (0) 92 3663 50 30 Fax +55 (0) 92 3663 50 27 filial.am@sew.com.br
	Mato Grosso/ Cuiabá	SEW-EURODRIVE Brasil Ltda. Avenida Miguel Sutil, 5573 Santa Helena - 78015-100	Tel. +55 (0) 65 3621 21 15 Fax +55 (0) 65 3621 64 31 filial.mt@sew.com.br
	Goiás Goiânia	SEW-EURODRIVE Brasil Ltda. Rua CM 14 - QD 03B - LT 06 Setor Cândida de Morais - 74463-280	Tel. +55 (0) 62 3299 54 44 Fax +55 (0) 62 3597 24 88 filial.go@sew.com.br
	Espírito Santo Serra	SEW-EURODRIVE Brasil Ltda. Rua Tancredo Neves, 88 Jardim Limoeiro - 29164-000	Tel. +55 (0) 27 33 18 09 21 Fax +55 (0) 27 33 18 09 25 filial.es@sew.com.br
	Endereços adicionais para Service no Brasil, fornecidos sob consulta!		

África do Sul			
Montadora Vendas / Service	**Johannesburg**	SEW-EURODRIVE (PROPRIETARY) LIMITED Johannesburg	Tel. +27 11 248-7000 dross@sew.co.za
	Capetown	SEW-EURODRIVE (PROPRIETARY) LIMITED Cape Town	Tel. +27 21 552-9820 dswanepoel@sew.co.za
	Durban	SEW-EURODRIVE (PROPRIETARY) LIMITED Durban	Tel. +27 31 700-3451 dtait@sew.co.za

Alemanha			
Administração / Fábrica / Vendas	**Bruchsal**	SEW-EURODRIVE GmbH & Co KG Bruchsal	Tel. +49 7251 75-0 http://www.sew-eurodrive.de
Service	**Central** Redutor / Motor	SEW-EURODRIVE GmbH & Co KG Graben-Neudorf	Tel. +49 7251 75-1710 sc-mitte-gm@sew-eurodrive.de
	Central Eletrônicos	SEW-EURODRIVE GmbH & Co KG Bruchsal	Tel. +49 7251 75-1780 sc-mitte-e@sew-eurodrive.de
	Norte	SEW-EURODRIVE GmbH & Co KG Garbsen (próximo a Hannover)	Tel. +49 5137 8798-30 sc-nord@sew-eurodrive.de
	Leste	SEW-EURODRIVE GmbH & Co KG Meerane (próximo a Zwickau)	Tel. +49 3764 7606-0 sc-ost@sew-eurodrive.de
	Sul	SEW-EURODRIVE GmbH & Co KG Kirchheim (próximo a München)	Tel. +49 89 909552-10 sc-sued@sew-eurodrive.de
	Oeste	SEW-EURODRIVE GmbH & Co KG Langenfeld (próximo a Düsseldorf)	Tel. +49 2173 8507-30 sc-west@sew-eurodrive.de
	Endereços adicionais para service na Alemanha, fornecidos sob consulta!		

Argélia			
Vendas	**Alger**	Réducom Alger	Tel. +213 21 8222-84

Argentina			
Montadora Vendas / Service	**Buenos Aires**	SEW EURODRIVE ARGENTINA S.A. Garin	Tel. +54 3327 4572-84 sewar@sew-eurodrive.com.ar

Austrália			
Montadora Vendas / Service	**Melbourne**	SEW-EURODRIVE PTY. LTD. Tullamarine, Victoria	Tel. +61 3 9933-1000 http://www.sew-eurodrive.com.au
	Sydney	SEW-EURODRIVE PTY. LTD. New South Wales	Tel. +61 2 9725-9900 enquires@sew-eurodrive.com.au

Áustria			
Montadora Vendas / Service	**Wien**	SEW-EURODRIVE Ges.m.b.H. Wien	Tel. +43 1 617 55 00-0 http://sew-eurodrive.at

Bélgica			
Montadora Vendas / Service	**Brüssel**	SEW Caron-Vector S.A. Wavre	Tel. +32 10 231-311 http://www.caron-vector.be

Bulgária			
Vendas	**Sofia**	BEVER-DRIVE GmbH Sofia	Tel. +359 2 9151160 bever@fastbg.net

Camarões			
Vendas	**Douala**	Electro-Services Douala	Tel. +237 4322-99

Canadá			
Montadora Vendas / Service	**Toronto**	SEW-EURODRIVE CO. OF CANADA LTD. Bramalea, Ontario	Tel. +1 905 791-1553 http://www.sew-eurodrive.ca

Canadá			
	Vancouver	SEW-EURODRIVE CO. OF CANADA LTD. Delta. B.C.	Tel. +1 604 946-5535 b.wake@sew-eurodrive.ca
	Montreal	SEW-EURODRIVE CO. OF CANADA LTD. LaSalle, Quebec	Tel. +1 514 367-1124 a.peluso@sew-eurodrive.ca
	Endereços adicionais para service no Canadá, fornecidos sob consulta!		
Chile			
Montadora **Vendas / Service**	**Santiago de Chile**	SEW-EURODRIVE CHILE LTDA. Santiago - Chile	Tel. +56 2 75770-00 ventas@sew-eurodrive.cl
China			
Montadora **Vendas / Service**	**Tianjin**	SEW-EURODRIVE (Tianjin) Co., Ltd. Tianjin	Tel. +86 22 25322612 http://www.sew.com.cn
Montadora **Vendas / Service**	**Suzhou**	SEW-EURODRIVE (Suzhou) Co., Ltd. Jiangsu Province	Tel. +86 512 62581781 suzhou@sew.com.cn
	Endereços adicionais para service na China, fornecidos sob consulta!		
Colômbia			
Montadora **Vendas / Service**	**Bogotá**	SEW-EURODRIVE COLOMBIA LTDA. Santafé de Bogotá	Tel. +57 1 54750-50 sewcol@sew-eurodrive.com.co
Coréia			
Montadora **Vendas / Service**	**Ansan-City**	SEW-EURODRIVE KOREA CO., LTD. Ansan	Tel. +82 31 492-8051 master@sew-korea.co.kr
Costa do Marfim			
Vendas	**Abidjan**	SICA Ste industrielle et commerciale pour l'Afrique Abidjan	Tel. +225 2579-44
Croácia			
Vendas **Service**	**Zagreb**	KOMPEKS d. o. o. Zagreb	Tel. +385 1 4613-158 kompeks@net.hr
Dinamarca			
Montadora **Vendas / Service**	**Kopenhagen**	SEW-EURODRIVEA/S Greve	Tel. +45 43 9585-00 http://www.sew-eurodrive.dk
Eslováquia			
Vendas	**Bratislava**	SEW-Eurodrive SK s.r.o. Bratislava	Tel. +421 2 49595201 http://www.sew.sk
	Zilina	SEW-Eurodrive SK s.r.o. Zilina	Tel. +421 41 700 2513 sew@sew-eurodrive.sk
	Banská Bystrica	SEW-Eurodrive SK s.r.o. Banská Bystrica	Tel. +421 48 414 6564 sew@sew-eurodrive.sk
Eslovênia			
Vendas **Service**	**Celje**	Pakman - Pogonska Tehnika d.o.o. Celje	Tel. +386 3 490 83-20 pakman@siol.net
Espanha			
Montadora **Vendas / Service**	**Bilbao**	SEW-EURODRIVE ESPAÑA, S.L. Zamudio (Vizcaya)	Tel. +34 9 4431 84-70 sew.spain@sew-eurodrive.es
Estonia			
Vendas	**Tallin**	ALAS-KUUL AS Tallin	Tel. +372 6593230 veiko.soots@alas-kuul.ee
EUA			
Montadora **Vendas / Service**	**Greenville**	SEW-EURODRIVE INC. Lyman	Tel. +1 864 439-7537 http://www.seweurodrive.com

EUA			
Montadora Vendas / Service	**San Francisco**	SEW-EURODRIVE INC. Hayward, California	Tel. +1 510 487-3560 cshayward@seweurodrive.com
	Philadelphia/PA	SEW-EURODRIVE INC. Bridgeport, New Jersey	Tel. +1 856 467-2277 csbridgeport@seweurodrive.com
	Dayton	SEW-EURODRIVE INC. Troy, Ohio	Tel. +1 937 335-0036 cstroy@seweurodrive.com
	Dallas	SEW-EURODRIVE INC. Dallas, Texas	Tel. +1 214 330-4824 csdallas@seweurodrive.com
	Endereços adicionais para service nos EUA, fornecidos sob consulta!		

Finlândia			
Montadora Vendas / Service	**Lahti**	SEW-EURODRIVE OY Hollola	Tel. +358 201 589-300 http://www.sew-eurodrive.fi

França			
Produção Vendas / Service	**Haguenau**	SEW-USOCOME Haguenau Cedex	Tel. +33 3 88 73 67 00 http://www.usocome.com
Montadora Vendas / Service	**Bordeaux**	SEW-USOCOME Pessac Cedex	Tel. +33 5 57 26 39 00
	Lyon	SEW-USOCOME Vaulx en Velin	Tel. +33 4 72 15 37 00
	Paris	SEW-USOCOME Verneuil l'Etang	Tel. +33 1 64 42 40 80
	Endereços adicionais para service na França, fornecidos sob consulta!		

Gabão			
Vendas	**Libreville**	Electro-Services Libreville	Tel. +241 7340-11

Grã-Bretanha			
Montadora Vendas / Service	**Normanton**	SEW-EURODRIVE Ltd. GB-Normanton, West- Yorkshire	Tel. +44 1924 893-855 http://www.sew-eurodrive.co.uk

Grécia			
Vendas Service	**Athen**	Christ. Boznos & Son S.A. Piraeus	Tel. +30 2 1042 251-34 http://www.boznos.gr

Holanda			
Montadora Vendas / Service	**Rotterdam**	VECTOR Aandrijftechniek B.V. Rotterdam	Tel. +31 10 4463-700 http://www.vector.nu

Hong Kong			
Montadora Vendas / Service	**Hong Kong**	SEW-EURODRIVE LTD. Kowloon, Hong Kong	Tel. +852 2 7960477 + 79604654 sew@sewhk.com

Hungria			
Vendas Service	**Budapest**	SEW-EURODRIVE Kft. Budapest	Tel. +36 1 437 06-58 office@sew-eurodrive.hu

Índia			
Montadora Vendas / Service	**Baroda**	SEW-EURODRIVE India Pvt. Ltd. Gujarat	Tel. +91 265 2831086 mdoffice@seweurodriveindia.com
Escritórios Técnicos	**Bangalore**	SEW-EURODRIVE India Private Limited Bangalore	Tel. +91 80 22266565 salesbang@seweurodriveinindia.com

Irlanda			
Vendas Service	**Dublin**	Alperton Engineering Ltd. Glasnevin, Dublin	Tel. +353 1 830-6277

Israel			
Vendas	**Tel-Aviv**	Liraz Handasa Ltd. Holon	Tel. +972 3 5599511 lirazhandasa@barak-online.net
Itália			
Montadora **Vendas / Service**	**Milano**	SEW-EURODRIVE di R. Blickle & Co.s.a.s. Solaro (Milano)	Tel. +39 02 96 9801 sewit@sew-eurodrive.it
Japão			
Montadora **Vendas / Service**	**Toyoda-cho**	SEW-EURODRIVE JAPAN CO., LTD Shizuoka	Tel. +81 538 373811 sewjapan@sew-eurodrive.co.jp
Letônia			
Vendas	**Riga**	SIA Alas-Kuul Riga	Tel. +371 7139386 info@alas-kuul.ee
Líbano			
Vendas	**Beirut**	Gabriel Acar & Fils sarl Bourj Hammoud, Beirut	Tel. +961 1 4947-86 gacar@beirut.com
Lituânia			
Vendas	**Alytus**	UAB Irseva Alytus	Tel. +370 315 79204 http://www.sew-eurodrive.lt
Luxemburgo			
Montadora **Vendas / Service**	**Brüssel**	CARON-VECTOR S.A. Wavre	Tel. +32 10 231-311 http://www.caron-vector.be
Macedônia			
Vendas	**Skopje**	SGS-Skopje / Macedonia Skopje / Macedonia	Tel. +389 2 385 466 sgs@mol.com.mk
Malásia			
Montadora **Vendas / Service**	**Johore**	SEW-EURODRIVE SDN BHD West Malaysia	Tel. +60 7 3549409 sales@sew-eurodrive.com.my
Marrocos			
Vendas	**Casablanca**	S. R. M. Société de Réalisations Mécaniques Casablanca	Tel. +212 2 6186-69 + 6186-70 + 6186-71 srm@marocnet.net.ma
México			
Montadora **Vendas / Service**	**Queretaro**	SEW-EURODRIVE, Sales and Distribution, S. A. de C. V. Queretaro, Mexico	Tel. +52 442 1030-300 scmexico@seweurodrive.com.mx
Noruega			
Montadora **Vendas / Service**	**Moss**	SEW-EURODRIVE A/S Moss	Tel. +47 69 241-020 sew@sew-eurodrive.no
Nova Zelândia			
Montadora **Vendas / Service**	**Auckland**	SEW-EURODRIVE NEW ZEALAND LTD. East Tamaki Auckland	Tel. +64 9 2745627 sales@sew-eurodrive.co.nz
	Christchurch	SEW-EURODRIVE NEW ZEALAND LTD. Christchurch	Tel. +64 3 384-6251 sales@sew-eurodrive.co.nz
Peru			
Montadora **Vendas / Service**	**Lima**	SEW DEL PERU MOTORES REDUCTORES Lima	Tel. +51 1 3495280 sewperu@sew-eurodrive.com.pe
Polônia			
Montadora **Vendas / Service**	**Lodz**	SEW-EURODRIVE Polska Sp.z.o.o. Lodz	Tel. +48 42 67710-90 http://www.sew-eurodrive.pl

Portugal			
Montadora **Vendas / Service**	**Coimbra**	SEW-EURODRIVE, LDA. Mealhada	Tel. +351 231 20 9670 http://www.sew-eurodrive.pt
República Tcheca			
Vendas	**Praha**	SEW-EURODRIVE CZ S.R.O. Vokovice	Tel. +420 a220121236 http://www.sew-eurodrive.cz
Romênia			
Vendas **Service**	**Bucuresti**	Sialco Trading SRL Bucuresti	Tel. +40 21 230-1328 sialco@sialco.ro
Rússia			
Montadora **Vendas / Service**	**St. Petersburg**	ZAO SEW-EURODRIVE St. Petersburg Russia	Tel. +7 812 3332522 +7 812 5357142 http://www.sew-eurodrive.ru
Senegal			
Vendas	**Dakar**	SENEMECA Dakar	Tel. +221 849 47-70 senemeca@sentoo.sn
Sérvia e Montenegro			
Vendas	**Beograd**	DIPAR d.o.o. Beograd	Tel. +381 11 3088677 / +381 11 3088678 dipar@yubc.net
Singapura			
Montadora **Vendas / Service**	**Singapore**	SEW-EURODRIVE PTE. LTD. Singapore	Tel. +65 68621701 sewsingapore@sew-eurodrive.com
Suécia			
Montadora **Vendas / Service**	**Jönköping**	SEW-EURODRIVE AB Jönköping	Tel. +46 36 3442-00 http://www.sew-eurodrive.se
Suíça			
Montadora **Vendas / Service**	**Basel**	Alfred Imhof A.G. Münchenstein bei Basel	Tel. +41 61 417 1717 http://www.imhof-sew.ch
Tailândia			
Montadora **Vendas / Service**	**Chon Buri**	SEW-EURODRIVE (Thailand) Ltd. Chon Buri	Tel. +66 38 454281 sewthailand@sew-eurodrive.co.th
Tunísia			
Vendas	**Tunis**	T. M.S. Technic Marketing Service Mégrine Erriadh	Tel. +216 1 4340-64 + 1 4320-29
Turquia			
Montadora **Vendas / Service**	**Istanbul**	SEW-EURODRIVE Maltepe ISTANBUL	Tel. +90 216 4419163 + 216 4419164 + 216 3838014 sew@sew-eurodrive.com.tr
Ucrânia			
Vendas **Service**	**Dnepropetrovsk**	SEW-EURODRIVE Dnepropetrovsk	Tel. +380 56 370 3211 sew@sew-eurodrive.ua
Venezuela			
Montadora **Vendas / Service**	**Valencia**	SEW-EURODRIVE Venezuela S.A. Valencia, Estado Carabobo	Tel. +58 241 832-9804 sewventas@cantv.net

Como movimentar o mundo

Com pessoas que pensam rapidamente e que desenvolvem o futuro com você.

Com a prestação de serviços integrados acessíveis a todo momento, em qualquer localidade.

Com sistemas de acionamentos e controles que potencializam automaticamente o seu desempenho.

Com o conhecimento abrangente nos mais diversos segmentos industriais.

Com elevados padrões de qualidade que simplificam a automatização de processos.

SEW-EURODRIVE
Solução em Movimento

Com idéias inovadoras que antecipam agora as soluções para o futuro.

Com uma rede global de soluções ágeis e especificamente desenvolvidas.

Com presença na internet, oferecendo acesso constante às mais novas informações, atualizações de softwares de aplicação.

SEW-EURODRIVE Brasil Ltda.
Avenida Amâncio Gaiolli, 50
Rodovia Presidente Dutra, Km 208
Guarulhos - 07251 250 - SP
SAT - SEW ATENDE - 0800 7700496
→ **www.sew-eurodrive.com**